Optimal Off-Line Stream Merging for Media-on-Demand

Amotz Bar-Noy
AT&T
Richard Ladner
U. of Washington

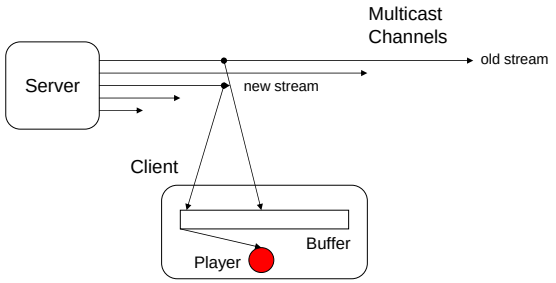
Media-On-Demand

- Two hour movie
- 5 minute song
- 2 minute news video clip

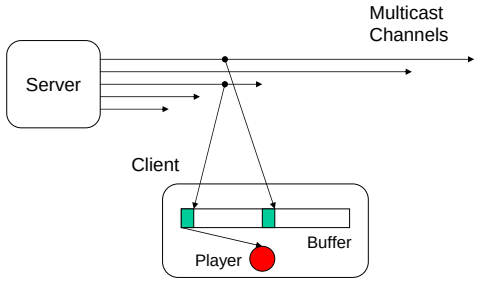
Goals for On-Demand Streams

- Satisfy the client
  - Minimize delay
  - Simplicity
- Reduce resource needs for the server
  - Minimize bandwidth
- Opportunities exist to satisfy the goals using Stream Merging if:
  - Adequate client receive bandwidth
  - Adequate client buffer storage
  - Multicast

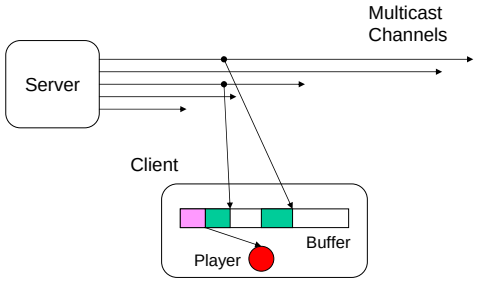
Stream Merging System



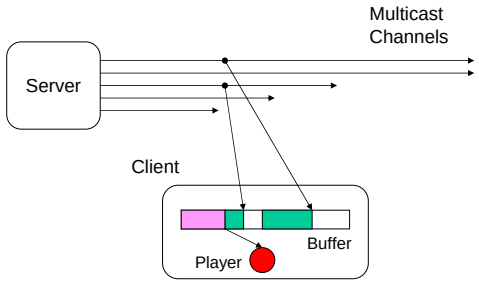
Stream Merging System



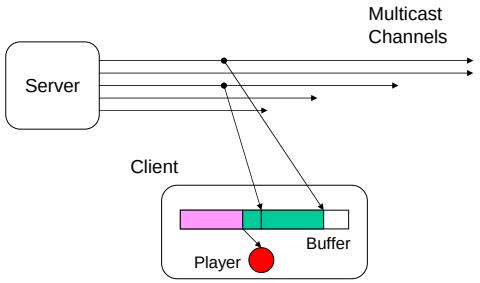
Stream Merging System



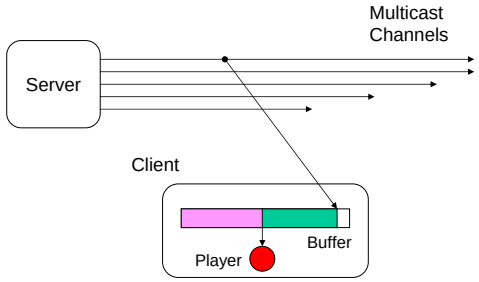
Stream Merging System



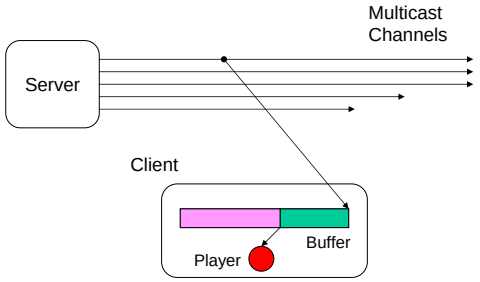
Stream Merging System



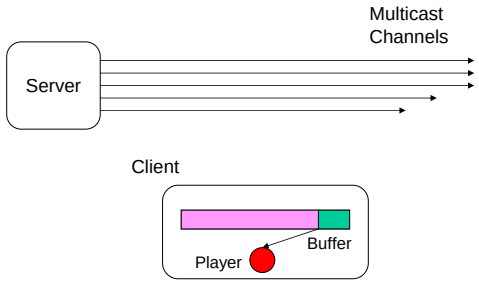
Stream Merging System



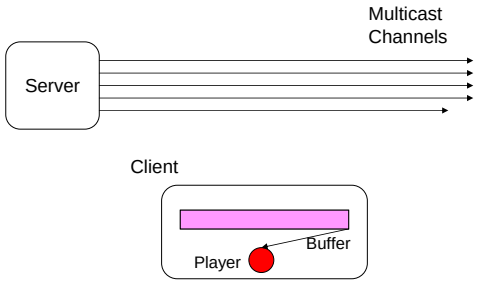
Stream Merging System



Stream Merging System

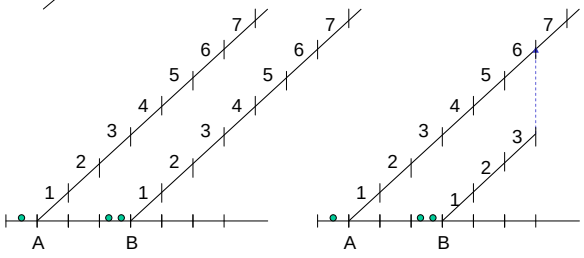


Stream Merging System



Batching vs. Stream Merging

client
delay guarantee
stream

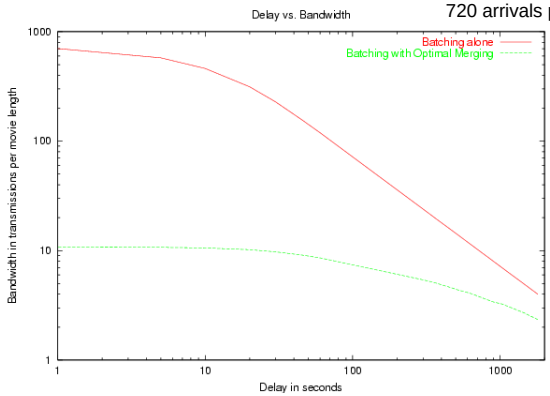


Batching

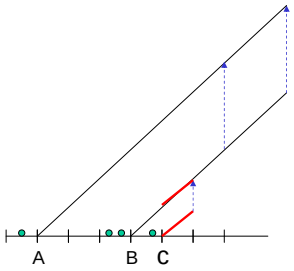
Batching with merging

Delay vs. Bandwidth Tradeoff

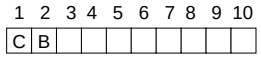
2 hour movie
720 arrivals per movie



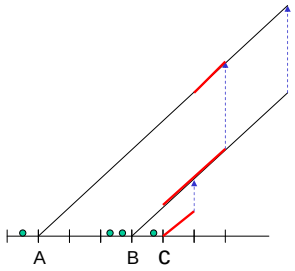
Clients C



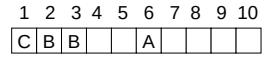
C: 1 - 1
B: 2 - 5
A: 6 - 10



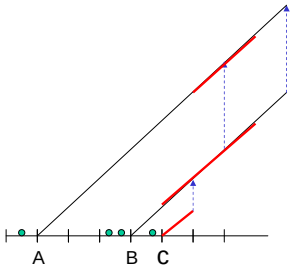
Clients C



C: 1 - 1
B: 2 - 5
A: 6 - 10



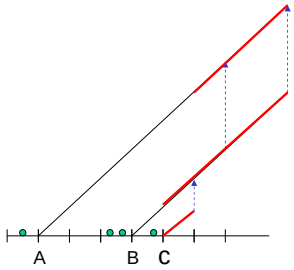
Clients C



C: 1 - 1
B: 2 - 5
A: 6 - 10

1	2	3	4	5	6	7	8	9	10
C	B	B	B		A	A			

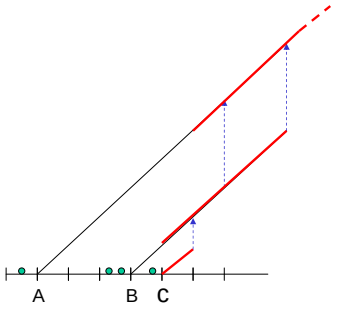
Clients C



C: 1 - 1
B: 2 - 5
A: 6 - 10

1	2	3	4	5	6	7	8	9	10
C	B	B	B	B	A	A	A		

Clients C



C: 1 - 1
B: 2 - 5
A: 6 - 10

1	2	3	4	5	6	7	8	9	10
C	B	B	B	B	A	A	A	A	A

Assumptions for Stream Merging

- Multicast of streams for simultaneous reception.
- Clients have receive bandwidth twice the playback bandwidth for each stream.
- Clients have adequate buffer space.
- In reply to a request clients are told once and for all which streams to listen to and when.

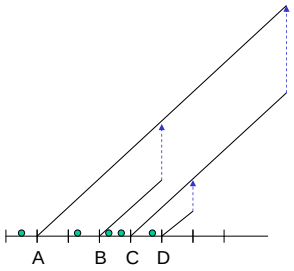
Off-line vs. On-line

- Off-line
  - Arrival sequence is known in advance
  - Fully loaded = delay guaranteed
- On-line
  - Future arrivals not known
  - Clients' behavior not affected by future arrivals
  - Server must add new streams and lengthen old streams to accommodate new arrivals

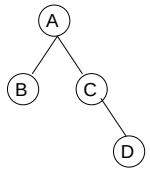
Contributions

- New model for stream merging
- Efficient optimal off-line algorithms
  - General case
  - Fully loaded
- Gain in using optimal off-line algorithm

The Merge Tree

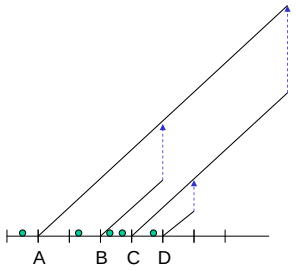


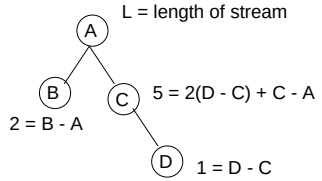
Concrete Merge Figure



Abstract Merge Tree

Calculating the Cost



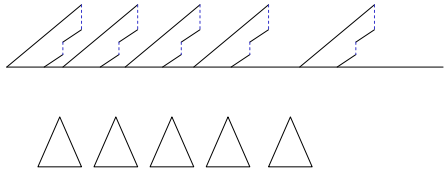


Merge cost = 8
Full cost = L + 8

- Root cost = length of stream
- Merge cost = sum of all streams except the root
- Full cost = root cost + merge cost

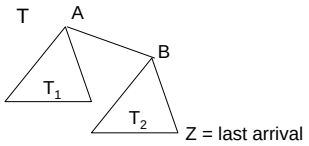
Merge Cost Focus

- Any general solution will be a merge forest.
  - A full stream has finite length



Off-line Optimal Merging

- Recursive formula for merge cost

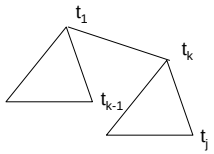


$MCost(T) = MCost(T_1) + MCost(T_2) + 2(Z - B) + (B - A)$

Dynamic Program for Optimal Merge Cost

- Setup
  - Arrivals $t_1, t_2, \ldots, t_n$
  - M(i,j) = minimum merge cost of a merge tree for the arrivals $t_i, t_2, \ldots, t_j$
  - M(1,n) is the optimal merge cost
- Recurrence
  - M(i,i) = 0
  - $M(i,j) = \min_{i < k \leq j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$
- $O(n^3)$ time, $O(n^2)$ storage
  - Aggarwal, Wolf, Yu (1996), Eager,Vernon, Zahorjan (1999)

The Recurrence



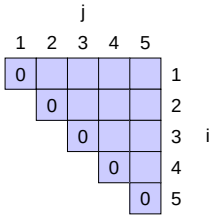
$M(i,j) = \min_{i < k \leq j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$

The Computation

i 1 2 3 4 5

t_i 0 3 4 5 8

j - i = 0



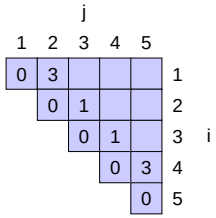
$M(i,j) = \min_{i < k \leq j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$

The Computation

i 1 2 3 4 5

t_i 0 3 4 5 8

j - i = 1



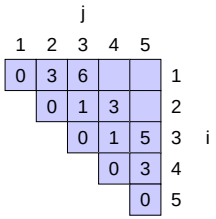
$M(i,j) = \min_{i < k \leq j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$

The Computation

i 1 2 3 4 5

t_i 0 3 4 5 8

j - i = 2



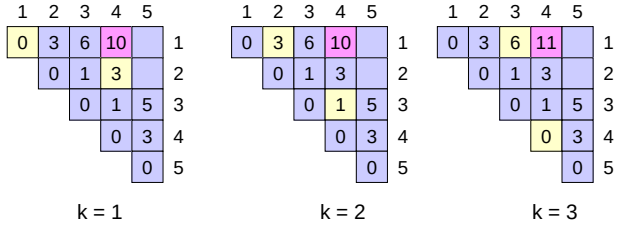
$M(i,j) = \min_{i < k \leq j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$

The Computation

i 1 2 3 4 5

t_i 0 3 4 5 8

j - i = 3



$M(i,j) = \min_{i < k \leq j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$

The Computation

i 1 2 3 4 5

t_i 0 3 4 5 8

j - i = 3

	1	2	3	4	5	
	0	3	6	10		1
		0	1	3	8	2
			0	1	5	3 i
				0	3	4
					0	5

$$M(i,j) = \min_{i < k \le j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$$

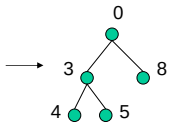
The Computation

i 1 2 3 4 5

t_i 0 3 4 5 8

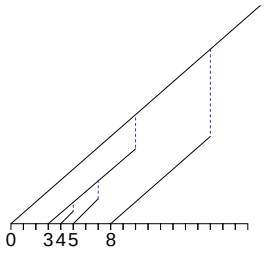
j - i = 4

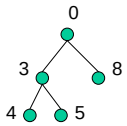
	1	2	3	4	5	
	0	3	6	10	18	1
		0	1	3	8	2
			0	1	5	3 i
				0	3	4
					0	5



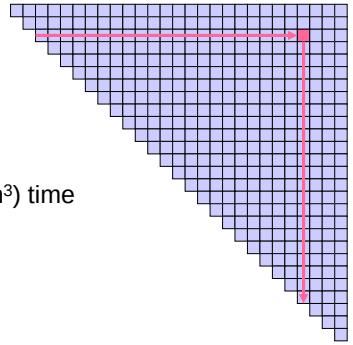
$$M(i,j) = \min_{i < k \le j}\{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$$

The Optimal Merge Tree





Optimal Algorithm Behavior



$O(n^3)$ time

$O(n^2)$ Optimal Algorithm

- Setup

  Arrivals $t_1, t_2, \ldots, t_n$

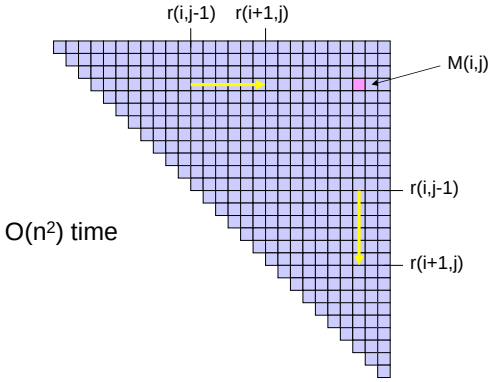
  r(i,j) = the right most stream that merges to the root of an optimal merge tree for the arrivals $t_i, \ldots, t_j$.

- Monotonicity (Knuth '71, F. Yao '80, Borchers, Gupta '94)

  $r(i,j-1) \leq r(i,j) \leq r(i+1,j)$

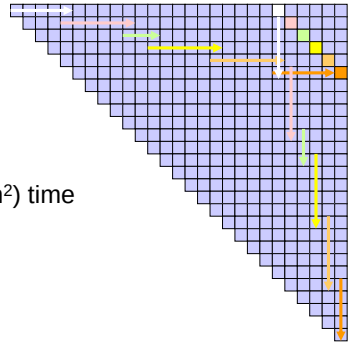
$$M(i,j) = \min_{r(i,j-1) \leq k \leq r(i+1,j)} \{M(i,k-1) + M(k,j) + 2(t_j - t_k) + (t_k - t_i)\}$$

New Algorithm Behavior



$O(n^2)$ time

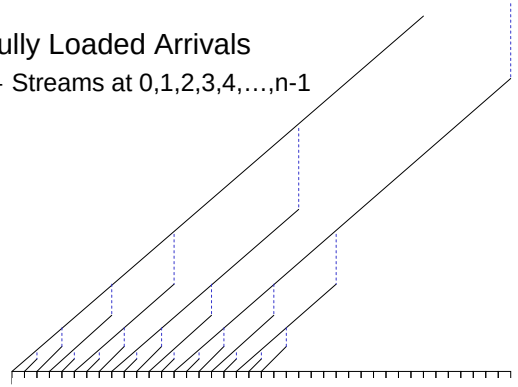
New Algorithm Behavior



$O(n^2)$ time

Fully Loaded Optimal Merge Tree

- Fully Loaded Arrivals
  - Streams at 0,1,2,3,4,…,n-1



Fibonacci Rules!

- Fibonacci numbers
  - 0,1,1, 2, 3, 5, 8,13, 21, 34, 55, 89, …
  - $F_0 = 0$, $F_1 = 1$, $F_k = F_{k-1} + F_{k-2}$
- Recurrence for fully loaded arrivals

$$M(1) = 0$$

$$M(n) = \min_{1 \le j \le n-1} \{M(j) + M(n-j) + 2n - j - 2\}$$

- Solution

$$M(n) = (k-1)n - F_{k+2} + 2, \quad F_k \le n \le F_{k+1}$$

Induction Hypothesis Design

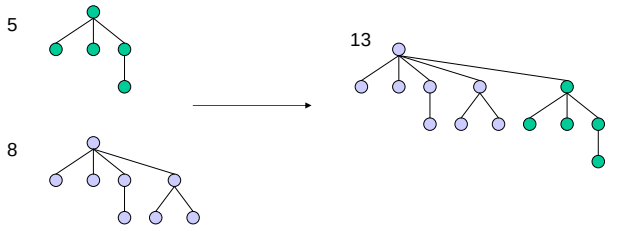
$$RM(n) = \arg\min_{1 \le j \le n-1} \{M(j) + M(n-j) + 2n - j - 2\}$$

2 {1},
3 {2},
4 {2, 3},
5 {3},
6 {3, 4},
7 {4, 5},
8 {5},
9 {5, 6},
10 {5, 6, 7},
11 {6, 7, 8},
12 {7, 8},
13 {8},
14 {8, 9}, 3
15 {8, 9, 10},
16 {8, 9, 10, 11}, 2
17 {9, 10, 11, 12},
18 {10, 11, 12, 13},
19 {11, 12, 13}, 3
20 {12, 13},

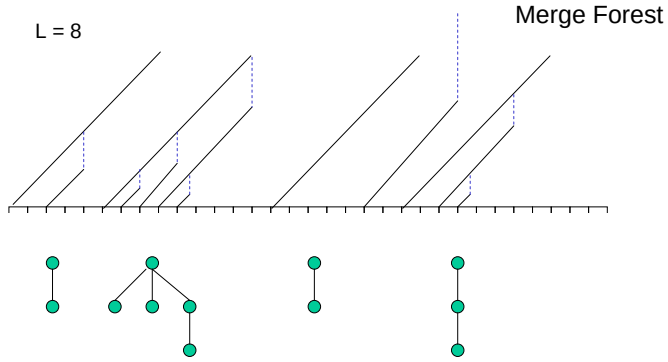
21 {13},
22 {13, 14},
23 {13, 14, 15}, 5
24 {13, 14, 15, 16},
25 {13, 14, 15, 16, 17},
26 {13, 14, 15, 16, 17, 18},
27 {14, 15, 16, 17, 18, 19}, 3
28 {15, 16, 17, 18, 19, 20},
29 {16, 17, 18, 19, 20, 21},
30 {17, 18, 19, 20, 21},
31 {18, 19, 20, 21}, 5
32 {19, 20, 21},
33 {20, 21},
34 {21},
35 {21, 22},
36 {21, 22, 23},
37 {21, 22, 23, 24}, 8
38 {21, 22, 23, 24, 25},
39 {21, 22, 23, 24, 25, 26},
40 {21, 22, 23, 24, 25, 26, 27},
41 {21, 22, 23, 24, 25, 26, 27, 28},
42 {21, 22, 23, 24, 25, 26, 27, 28, 29},
43 {22, 23, 24, 25, 26, 27, 28, 29, 30},
44 {23, 24, 25, 26, 27, 28, 29, 30, 31}, 5
45 {24, 25, 26, 27, 28, 29, 30, 31, 32},
46 {25, 26, 27, 28, 29, 30, 31, 32, 33},
47 {26, 27, 28, 29, 30, 31, 32, 33, 34},
48 {27, 28, 29, 30, 31, 32, 33, 34},
49 {28, 29, 30, 31, 32, 33, 34},
50 {29, 30, 31, 32, 33, 34},
51 {30, 31, 32, 33, 34},
52 {31, 32, 33, 34}, 8
53 {32, 33, 34},
54 {33, 34},

55 {34},
56 {34, 35},
57 {34, 35, 36},
58 {34, 35, 36, 37},
59 {34, 35, 36, 37, 38},
60 {34, 35, 36, 37, 38, 39},
61 {34, 35, 36, 37, 38, 39, 40}, 13
62 {34, 35, 36, 37, 38, 39, 40, 41},
63 {34, 35, 36, 37, 38, 39, 40, 41, 42},
64 {34, 35, 36, 37, 38, 39, 40, 41, 42, 43},
65 {34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44},
66 {34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45},
67 {34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46},
68 {34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47},
69 {35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48},
70 {36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49},
71 {37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50}, 8
72 {38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51},
73 {39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52},
74 {40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53},
75 {41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54},
76 {42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55},
77 {43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55},
78 {44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55},
79 {45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55},
80 {46, 47, 48, 49, 50, 51, 52, 53, 54, 55},
81 {47, 48, 49, 50, 51, 52, 53, 54, 55},
82 {48, 49, 50, 51, 52, 53, 54, 55},
83 {49, 50, 51, 52, 53, 54, 55}, 13
84 {50, 51, 52, 53, 54, 55},
85 {51, 52, 53, 54, 55},
86 {52, 53, 54, 55},
87 {53, 54, 55},
88 {54, 55},
89 {55},

Fully Loaded Optimal Trees



Full Cost

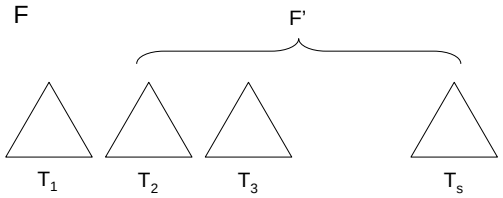


Optimal Full Cost

$$FCost(F) = s \cdot L + \sum_{i=1}^{s} MCost(T_i)$$

Find the merge forest F that minimizes Fcost(F).

Recursive Definition of Full Cost



$$FCost(F) = L + MCost(T_1) + FCost(F')$$

Optimal Full Cost Algorithm

- Setup
  - Arrivals $t_1, t_2, \ldots, t_n$
  - G(i) = the optimal full cost for the last n-i+1 arrivals $t_i, t_{i+1}, \ldots, t_n$
  - G(1) is the optimal full cost
- Recurrence
  - G(n+1) = 0
  - G(i) = L + min{ M(i,k-1) + G(k) : $i < k \leq n+1$ and $t_{k+1} - t_i \leq L-1$}

Full Cost for Fully Loaded

- The optimization simplifies
- O(n) time to compute the optimal merge forest

Gain in Using Stream Merging

- For fully loaded
  - $\log_\phi L / L$ reduction in bandwidth
- Example: 2 hour movie shown every minute
  - $L = 120$
  - $\log_\phi L / L = .083$
  - Server can show 12 different movies instead of 1.

Conclusions

- Stream Merging can be effective in reducing bandwidth
- Optimal off-line stream merging is efficient
- Optimal fully loaded stream merging is even more efficient