CMPT 886
Stream Merging Algorithms

Akiko Campbell

Presentation -1

Summer/2004

Agenda

- Background
- Basic Mechanism
- Performance Comparisons
- Representative Algorithms
  - Fibonacci Tree Algorithm
  - Dyadic Tree Algorithm
- Extension
- Conclusion

Background

- What is Stream Merging?
  - Technique to deliver media in multiple streams, using <u>multicast</u> and <u>client buffers</u>.
- Basic Idea
  - Medium: partitioned into equally-sized units.
  - Server: sends each client a “Client Receive Program”.
  - Clients: pre-receive and store data in buffers.
  - Streams: terminated asap.
- Objective
  - Reduce server bandwidth by having clients receive two or more streams simultaneously and terminate streams as soon as there are no needs for them.

Background

- Properties
  - On-line
  - Zero Delay
  - Receive-Two Model
  - Client Buffer size: up to half the medium

Background

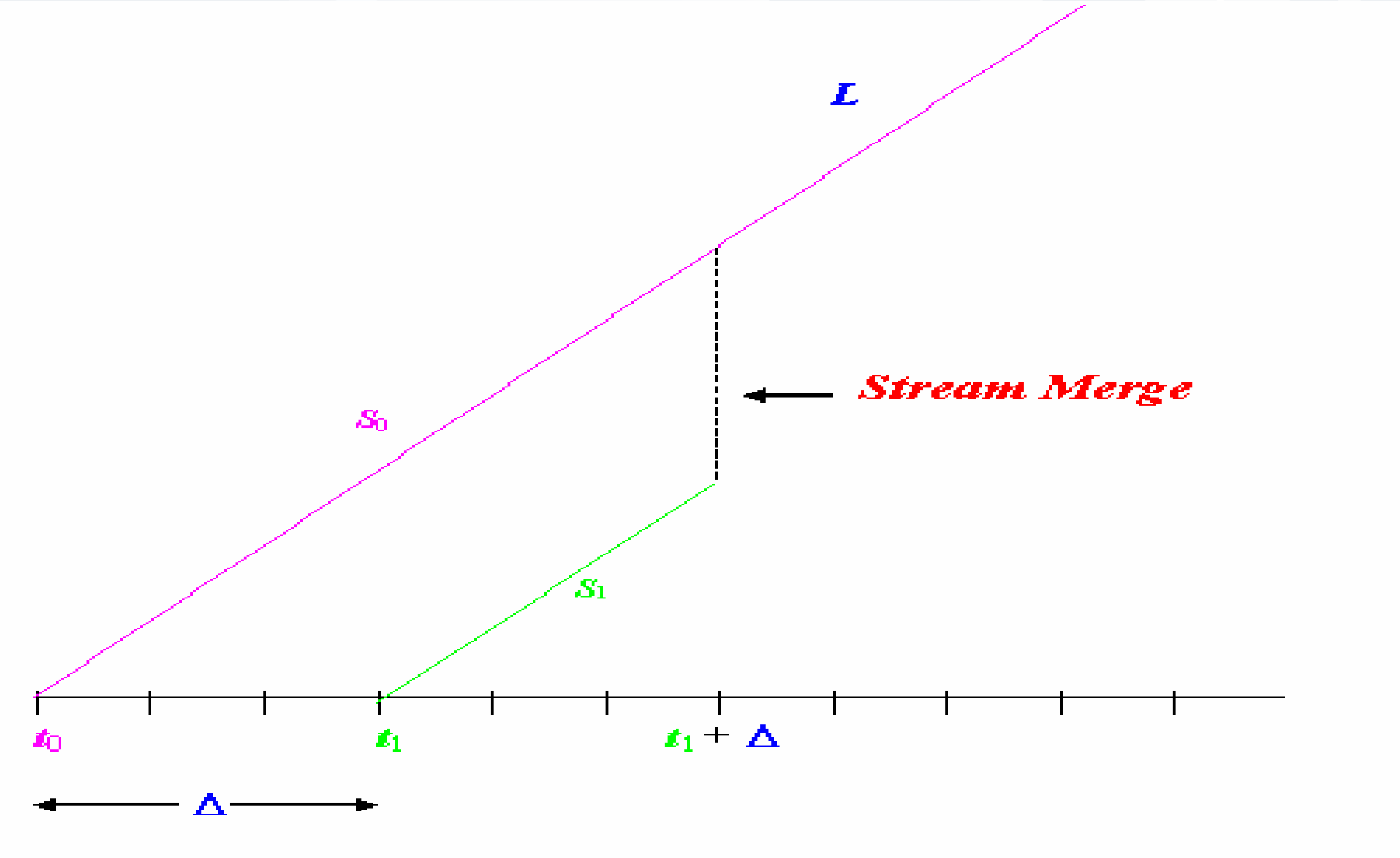
- Stream Merging Models
  - Merge Tree Based Model
    - Fibonacci Tree Algorithm
    - Dyadic Tree Algorithm
      - 2-dyadic
      - ϕ-dyadic
  - Event-Driven Model

Basic Mechanism

- Idea
  - Client C_1 arrives at time t_1 and requests a video of duration L.
  - Client C_0 is currently playing the same video from stream S_0 that began at time $t_0 < t_1$.
  - C_1 missed the first $\Delta = t_1 - t_0$ time units.
  - C_1 receives Δ from S_1.
  - At the same time, C_1 receives $L - \Delta$ from S_0.
  - Terminate S_1 after Δ time units (*Stream Merge*).

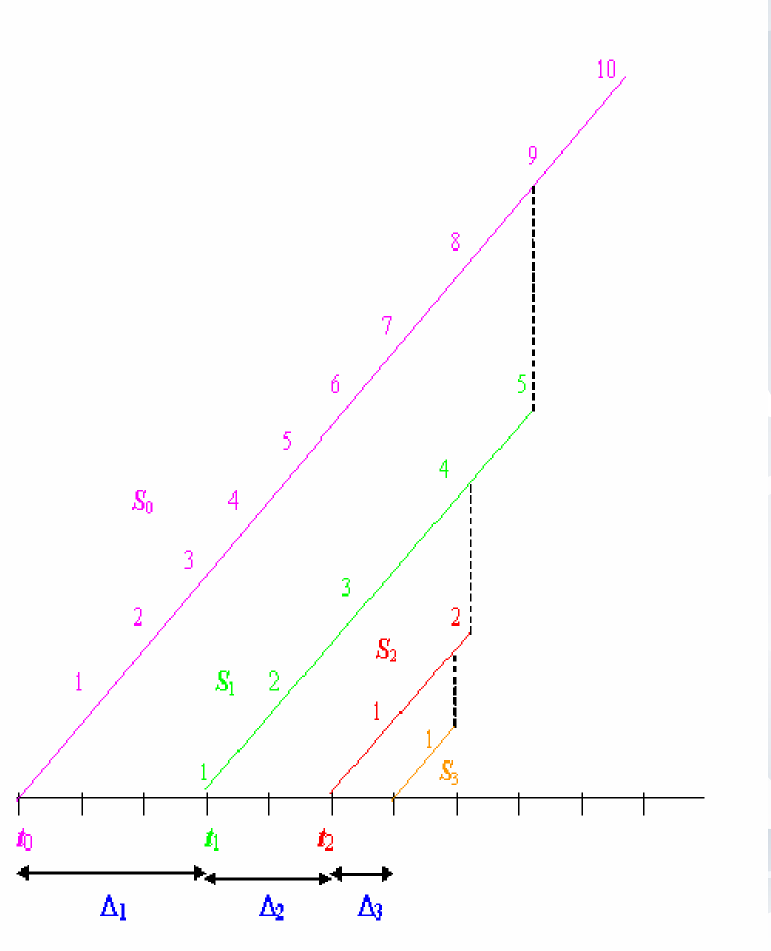
Basic Mechanism

- Concrete Merge Diagram



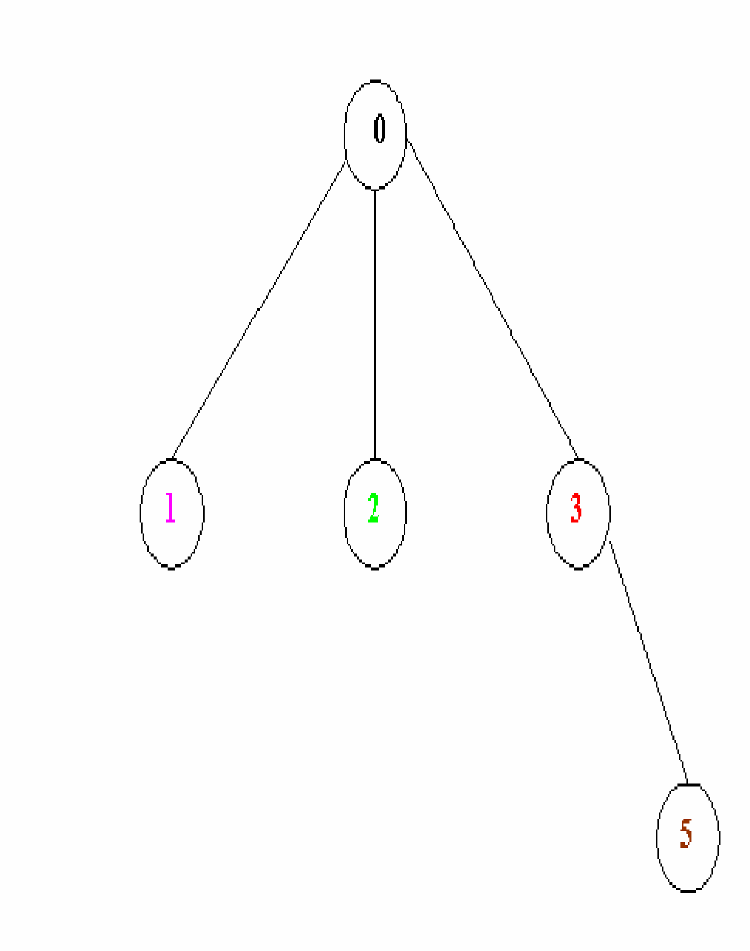
Basic Mechanism

- Total Data Capacity Required
  - Sum all stream lengths
  - Example:
    - $S_0 = 10$ for C_0
    - $S_1 = 5$ for C_1
    - $S_2 = 2$ for C_2
    - $S_3 = 1$ for C_3
  - ***Total Capacity*** = 18



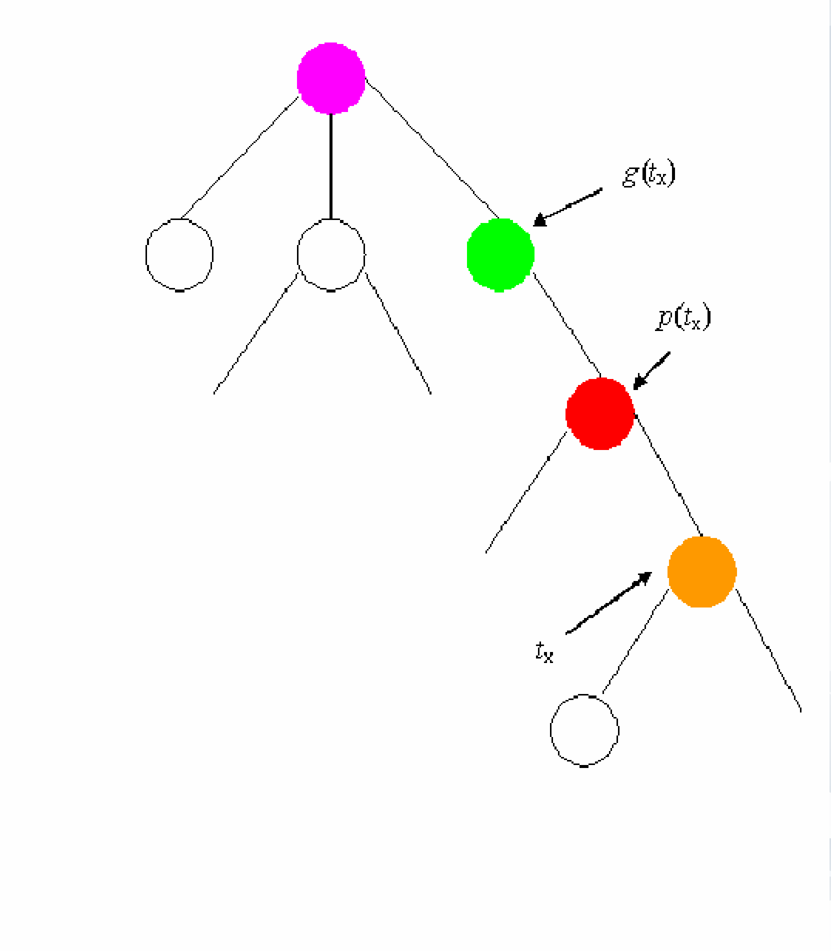
Basic Mechanism

- Merge Tree
  - Alternative to Concrete Merge Diagram.
  - Each node is labeled with client's arrival time.
  - Root: full stream length
  - Example:
    - C_1 arrived at time 1
    - C_2 arrived at time 2
    - C_3 arrived at time 3
    - C_4 arrived at time 5



Basic Mechanism

- Merge Tree
  - Client C_x receives from stream started at times t_x and its parent $p(t_x)$.
  - Stream started at t_x terminates.
  - C_x continues to receive from $p(t_x)$ and grandparent $g(t_x)$.



Basic Mechanism

- Merge Tree: Determining performance…

Total Data Capacity Required = Sum of All Stream Lengths

- Length of a Stream: $\boldsymbol{l(x)}$

$$\text{Non-leaf:}\quad \boldsymbol{l(x) = 2\,z(x) - x - p(x)}$$
$$\text{Leaf:}\quad \boldsymbol{l(x) = x - p(x)}$$

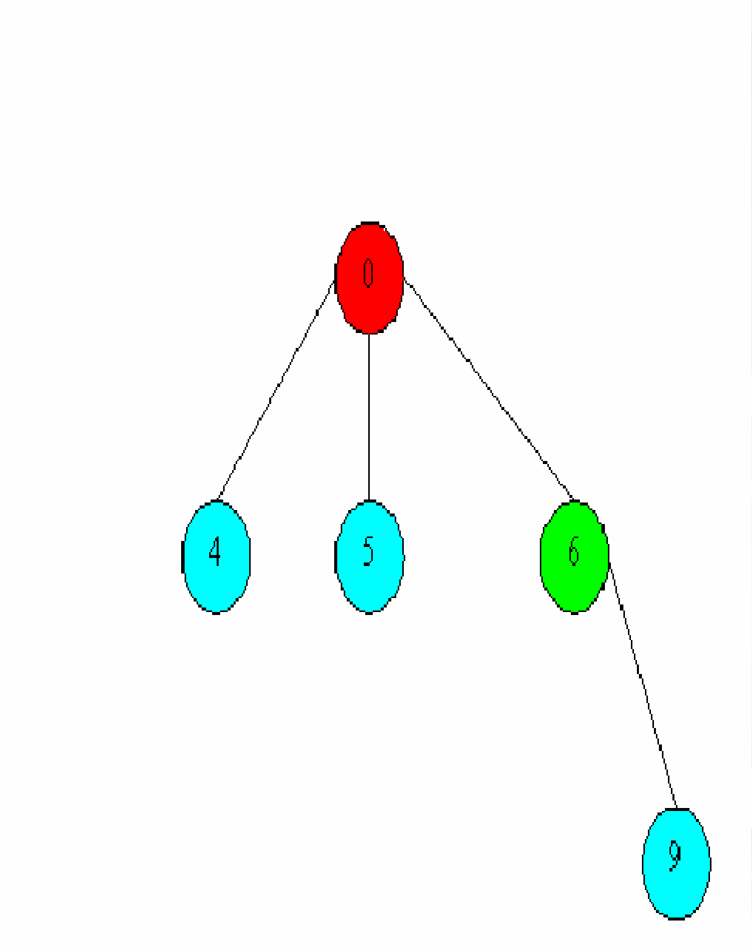
- **Where**
  - **$p(x)$ = parent of node x**
  - **$z(x)$ = the latest node in the sub-tree rooted at x**

Performance Comparisons

- How to Merge?
  - Merge decisions affect performance…
  - Example Scenario
    - Medium Length: 20
    - Arrival Times: $t = [0, 4, 5, 6, 9]$
  - Total Capacity?
    - Fibonacci: 44
    - 2-dyadic: 42
    - ϕ-dyadic: 40

Performance Comparisons

- Fibonacci: 44
  - *l*(Root): 20
  - Σ *l*(Non Leaf): 12
    2*9 – 6 – 0
  - Σ *l*(Leaf): 12
    (4-0) + (5-0) + (9-6)
- Total Capacity: 44

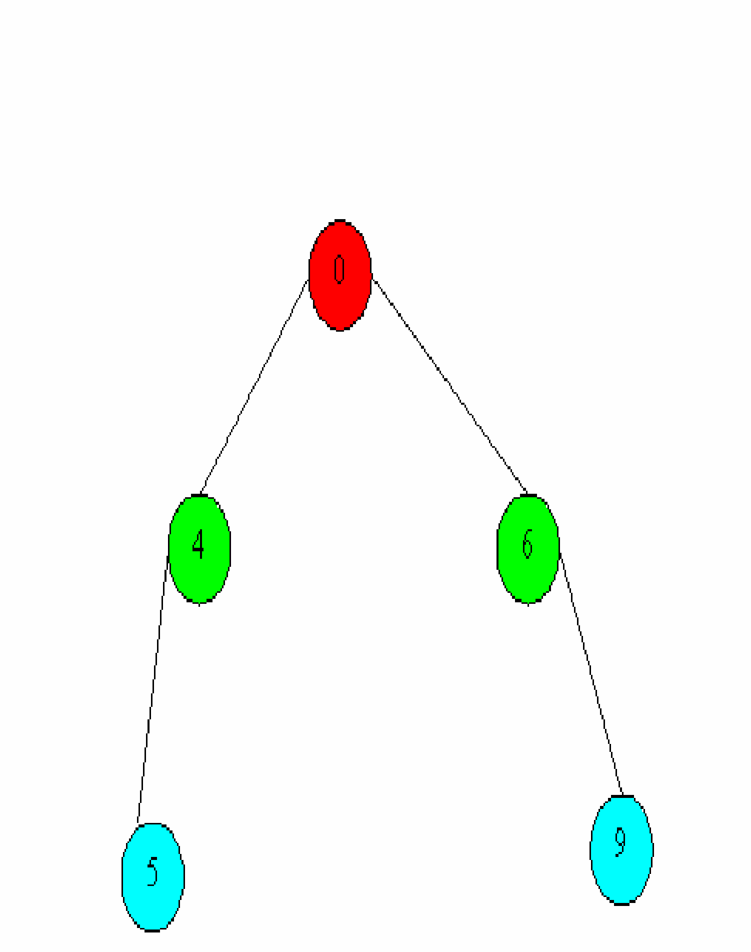


Performance Comparisons

- 2-dyadic: 42
  - *l*(Root): 20
  - Σ *l*(Non Leaf): 18

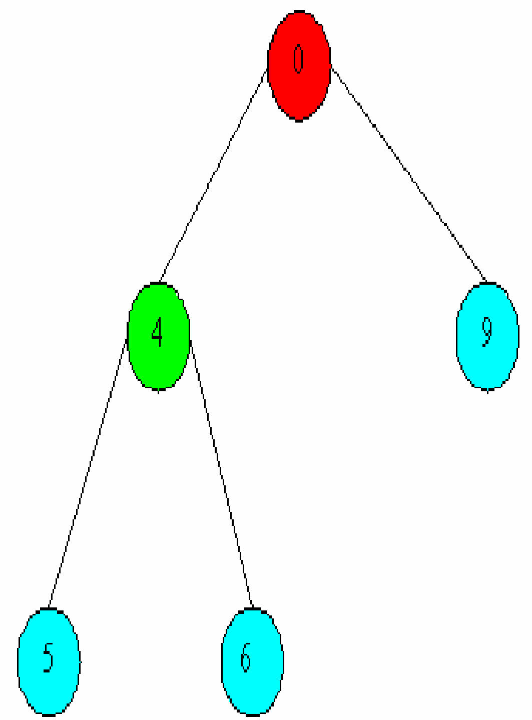
    (2*5 – 4 – 0) + (2*9 – 6 – 0)
  - Σ *l*(Leaf): 4

    (5-4) + (9-6)
- Total Capacity: 42



Performance Comparisons

- ϕ-dyadic: 40
  - l(Root): 20
  - $\Sigma\ l$(Non Leaf): 8
    2*6 – 4 – 0
  - $\Sigma\ l$(Leaf): 12
    (5-4) + (6-4) + (9-0)
- Total Capacity: 40



Performance Comparisons

- Shape of the tree determines the algorithm performance…
- What does the tree shape signify?
  - Merge Decisions.
  - Policy for creating Client Receive Program.
- How are the Tree Shapes determined?

Representative Algorithms

- Merge Tree Based Models
  - Fibonacci Tree Algorithm – worst case analysis
  - Dyadic Algorithm – average case analysis

Representative Algorithms

- Fibonacci vs. Dyadic
  - Common:
    - Based on Merge Trees.
    - Stream lengths can change at any time.
    - No change in Client Receive Program.
    - Heuristically start a new root.
  - Difference:
    - Policy for creating Client Receive Program.

Representative Algorithms

Fibonacci Tree Algorithm

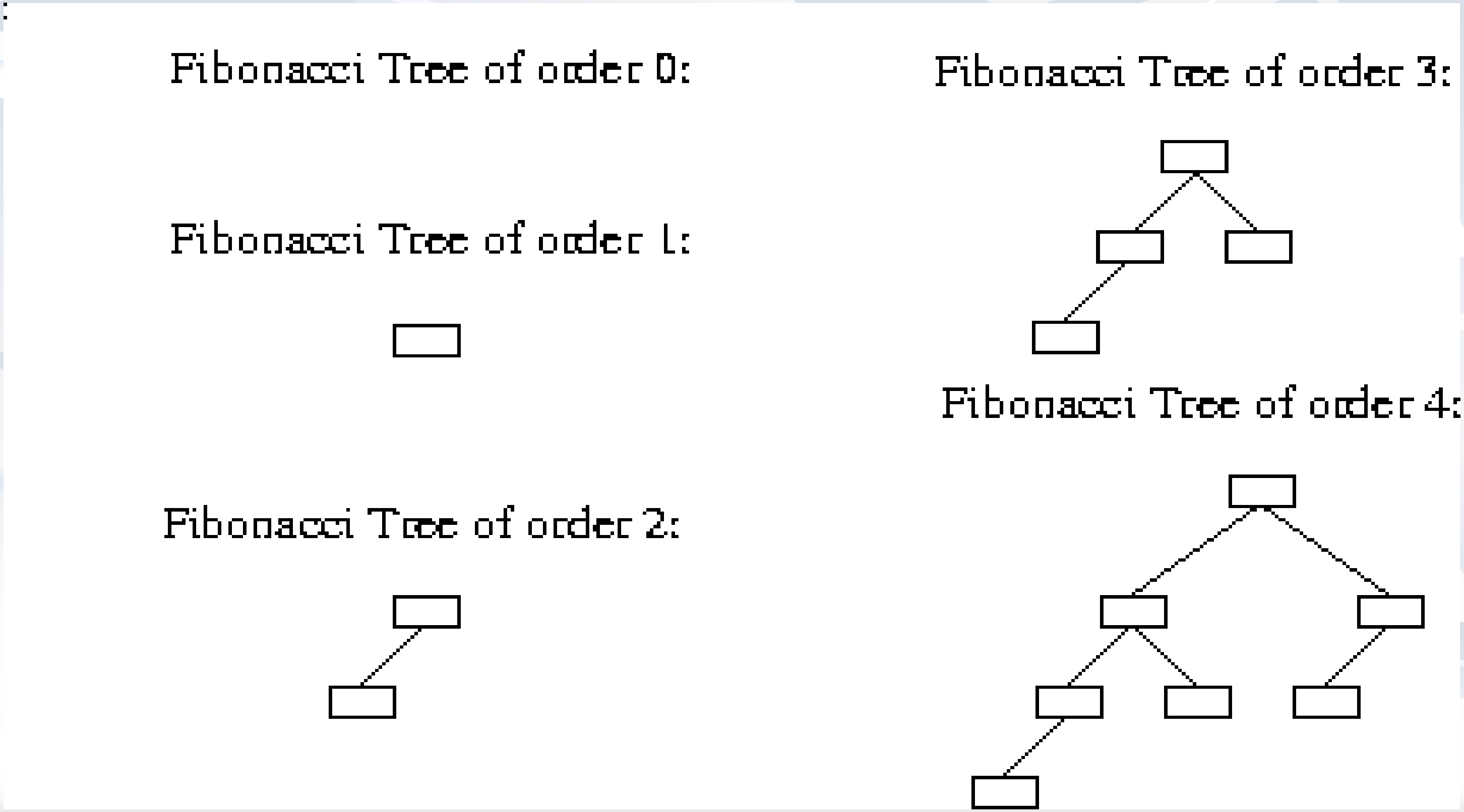
- Fibonacci Tree Refresher: formal definition

  A *Fibonacci tree* is a variety of binary tree restricted to have a special structure, which is defined recursively as:

  - The empty tree is a *Fibonacci tree of order 0*.
  - A tree with a single node is a *Fibonacci tree of order 1*.
  - For n > 1, a *Fibonacci tree of order n* consists of a root node, a left subtree that is a *Fibonacci tree of order n-1*, and a right subtree that is a *Fibonacci tree of order n-2*.

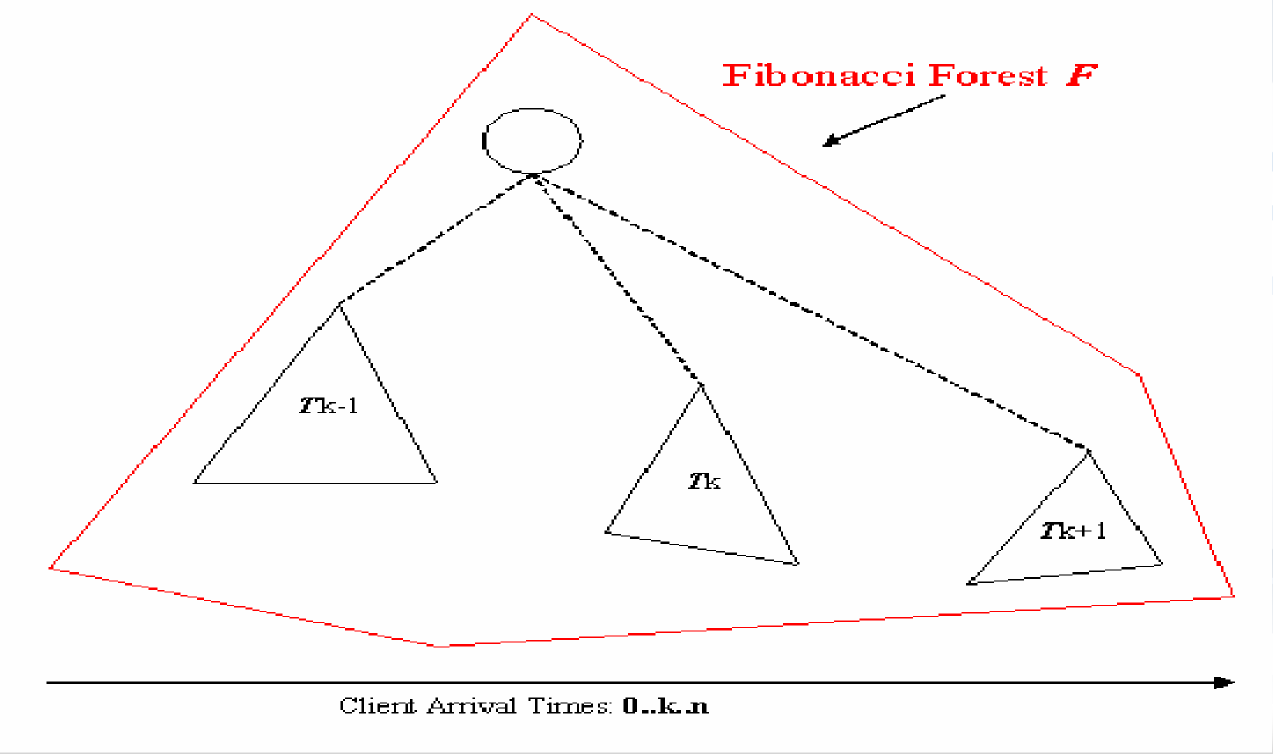
Representative Algorithms
Fibonacci Tree Algorithm

- Fibonacci Tree Refresher: Examples



Representative Algorithms
Fibonacci Tree Algorithm

- Fibonacci Tree properties
  - Infinite
  - Preorder



Representative Algorithms
Fibonacci Tree Algorithm

- What to do with the new arrival…
  - Add to an existing tree?
  - Root of a new tree?
- Start Rule
  - Client C_n arrives at t_n
    - Medium Length: ***L***
    - Forest: ***F***
    - t_m : root of the last tree in $\boldsymbol{F}_{n-1}$
  - If $t_n - t_m > L/2$: $t_n \rightarrow$ new root

Representative Algorithms
Fibonacci Tree Algorithm

- Start Rule Example
  - Medium Length: 20
  - Client arrival times: *t*[0, 4, 5, 6, 9, 11, 19]
  - Distance ($t_n - t_m$):
    - *t*[0]: → create a tree T_0
    - *t*[4, 5, 6, 9]: (4-0)…(9-0) < 20/2 → add to T_0
    - *t*[11]: (11-0) > 20/2 → new tree T_{11}
    - *t*[19]: (19-11) < 20/2 → add to T_{11}

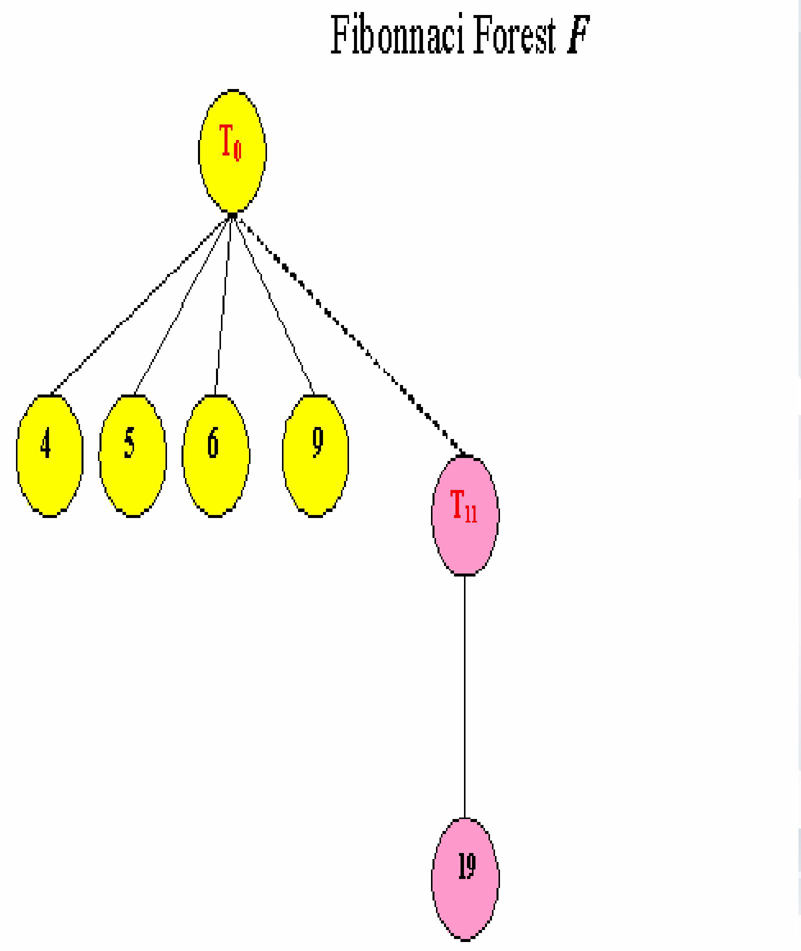
Representative Algorithms
Fibonacci Tree Algorithm

- Start Rule Example

  Fibonacci Forest ***F***

  - Create T_0.
  - Create T_{11} and connect with T_0.

- Merging Rules
  - Basic Merging Rule
  - Nearest Fit Rule
  - Best Fit Rule



Representative Algorithms
Fibonacci Tree Algorithm

- Merging Rules: *Basic Merging Rule*
  - Node to connect to: which node can be the parent of the new root?
  - Parameters
    - New root: t_n
    - Merge Tree: $\boldsymbol{F}_{n\text{-}1}$
    - $t_n + \boldsymbol{F}_{n\text{-}1} \rightarrow \boldsymbol{F}_n$
    - ***Right Frontier*** of $\boldsymbol{F}_{n\text{-}1}$: $t_1 = node_0, \ldots, node_i, \ldots, node_k = t_{n\text{-}1}$
  - $node_i$ on *Right Frontier* of $\boldsymbol{F}_{n\text{-}1} \rightarrow$ parent of t_n

Representative Algorithms

Fibonacci Tree Algorithm

- Merging Rules: *Basic Merging Rule*

  Not all nodes eligible parents…

  - S_i = Stream started at $node_i$
  - S_i ($i > 0$) terminated before t_n?
    - S_i terminates at $t_i + l(S_i)$.
    - $l(S_i) = 2t_k - t_i - t_{i\text{-}1} \rightarrow t_i + l(S_i) = t_i + (2t_k - t_i - t_{i\text{-}1})$
      $= 2t_k - t_{i\text{-}1}$
  - $node_i$ to be a parent…
    - $t_n \leq 2t_k - t_{i\text{-}1}$

Representative Algorithms

Fibonacci Tree Algorithm

- Merging Rules: *Basic Merging Rule*

  <u>*Right Frontier*</u> of $\boldsymbol{F}_{n\text{-}1}$: $t_1 = node_0 ,\ldots, node_i ,\ldots, node_k = t_{n\text{-}1}$

  - $node_i : i = 0$
    - $\boldsymbol{F}_n = \boldsymbol{F}^{\,0}_{\ n\text{-}1}$
  - $node_i : i > 0$
    - $\boldsymbol{F}_n = \boldsymbol{F}^{\,i}_{\ n\text{-}1} \,(t_n \leq 2\, t_{n\text{-}1} - t_{i\text{-}1})$

Representative Algorithms

Fibonacci Tree Algorithm

- Merging Rules: *Basic Merging Rule*
  - Basic Merging Rule
    - Property of the candidate parent.
    - But not how to pick a parent…

      → Nearest Fit Rule

      Best Fit Rule

Representative Algorithms
Fibonacci Tree Algorithm

- Merging Rules: *Nearest Fit Rule*
  - Pick a parent closest to the new arrival.
  - $node_i$: the parent of t_n where i is as large as possible.

    Right Frontier of $\boldsymbol{T}_{\text{n-1}}$: $t_1 = node_0, \ldots, node_i, \ldots, node_k = t_{n-1}$

    → Pick largest i $(1 \le i \le k)$ s.t. $t_n \le 2\, t_{n-1} - t_{i-1}$

Representative Algorithms

Fibonacci Tree Algorithm

- Merging Rules: *Best Fit Rule*
  - Pick a parent which minimizes the merge cost ***M*cost** of the resulting tree.
  - Merge Cost of tree *F*

$$\mathbf{\mathit{M}cost(\mathit{F}) = \sum_{x \in \{\mathit{F}\text{-root}\}} \mathit{l}(\mathit{x})}$$

Representative Algorithms

Fibonacci Tree Algorithm

- Building a Merge Tree: COST
  - $\boldsymbol{O}(n^2) \leftarrow$ Dynamic Programming + *Monotonicity*

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

$$\boldsymbol{M(i,i) = 0}$$

Arrivals : $t_1, t_2, \ldots, t_n$

$(i,k\text{-}1)$: Sub-tree of all arrivals prior to *k* (*incl* root(T))

(k,j): Sub-tree rooted at *k*

$(2t_j\text{-}t_k\text{-}t_i)$: length of the stream started at *k*

Representative Algorithms
Fibonacci Tree Algorithm

- How Many Channels?
  - Medium Length: L
  - Arrivals : $t_1, t_2, \ldots, t_n$
  - Span: $N = t_n - t_1$
  - Density ρ : n/N
    - Density of n arrivals over N

Representative Algorithms
Fibonacci Tree Algorithm

- Density $\rho : 0 < \rho \leq 1$
  - Worst Case: ρ close to 0
    - very few arrivals over N
    - Each arrival gets a full stream
    - Number of full Streams: $O(nL)$
  - Best Case: ρ close to 1
    - many merging activities
    - Number of full Streams: $O(N\log(\rho L))$ <<Detail in Appendix>>

Representative Algorithms

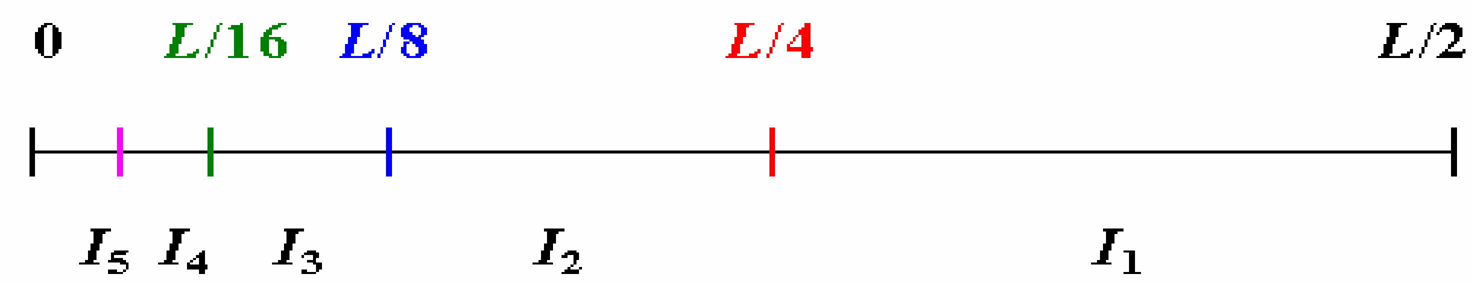
Dyadic Tree Algorithm

- 2-Dyadic Model: *Merge Rules*
  - Start Rule
    - Arrival Times: t_0 ,… t_i …, t_n (t_0 = 0)
      - $t_i \geq L/2 \rightarrow$ new tree
      - $t_i < L/2 \rightarrow$ child in dyadic intervals of t_0

Representative Algorithms

Dyadic Tree Algorithm

- 2-Dyadic Model: *Merge Rules*
  - Medium Length: L
  - Partition L into 2-dyadic intervals: I_i



Representative Algorithms

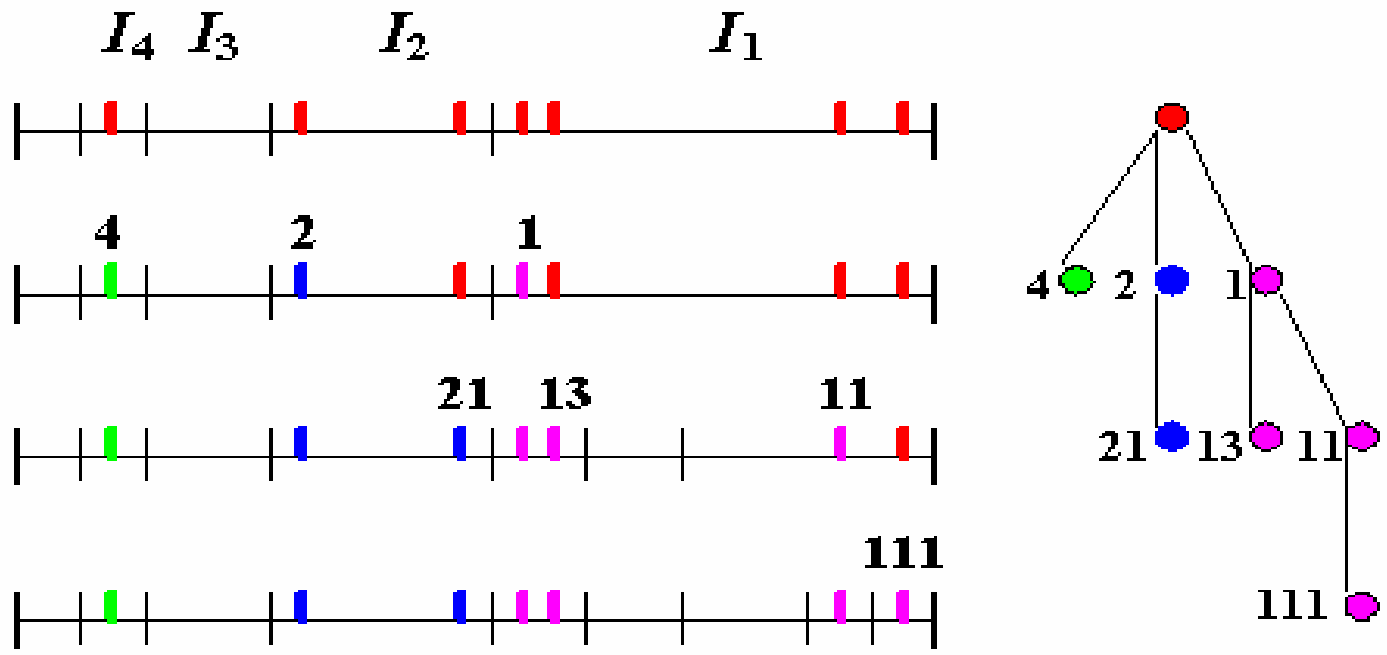
Dyadic Tree Algorithm

- 2-Dyadic Model: *Merge Rules*
  - Arrival Times: $t_0 ,\ldots\ t_i \ldots,\ t_n\ (t_0 = 0)$
  - If t = first arrival in I_i → label t with i.
  - If $t = k^{\text{th}}$ arrival in I_i → label t with ik.
  - Recursively label all arrivals

Representative Algorithms

Dyadic Tree Algorithm

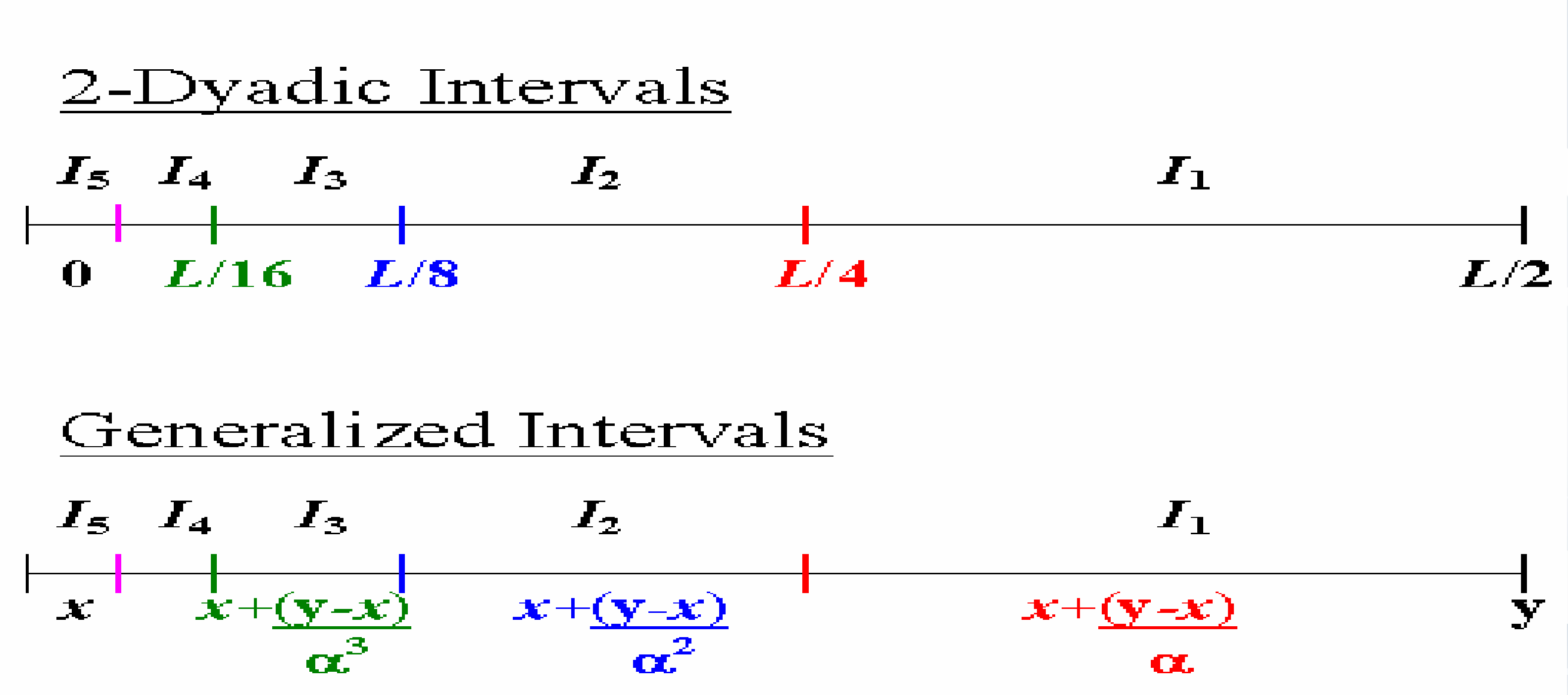
- 2-Dyadic Model: *Merge Rule Example*



Representative Algorithms

Dyadic Tree Algorithm

- ϕ-dyadic Model: *Generalized Intervals*



Representative Algorithms
Dyadic Tree Algorithm

- ϕ-dyadic Model: *Generalized Intervals*
  - Value of α
    - 2-Dyadic: 2
      - Coffman *et al*.
    - ϕ-Dyadic: $(1+\sqrt{5})/2$
      - Bar-Noy *et al*.
      - 20 hour experiment:
        - ***L***: 1-hour
        - Poisson with a mean interval arrival rate of 1 second
        - α: 1.2 – 2.4 in increment of 0.01
        - Best α = 1.65 (by means of the least # of streams)

Extension

- Stream Merging vs. Broadcasting
  - Common:
    - Reduce Server Bandwidth.
    - Minimize Start-up Delay.
    - Partition Medium into Sub-partitions.
    - Use of Multicast.
    - Use of Client Buffer.

Extension

- Stream Merging vs. Broadcasting
  - Major Difference:
    - Start-up Delay
      - Stream Merging: Zero Delay
        - Start a new stream for every new arrival.
      - Broadcasting: Some Delay (Fixed or Variable)

Extension

- Stream Merging vs. Broadcasting
  - Number of Client Arrivals:
    - Channel (Stream) Length
      - Stream Merging: Significant
      - Broadcasting: No Effect
    - Channel Allocation
      - Stream Merging: Significant
      - Broadcasting: No Effect

Extension

- Stream Merging vs. Broadcasting
  - Page Handling:
    - Page Sequence
      - Stream Merging: Consecutive Order
      - Broadcasting (e.g. Pagoda): Not Consecutive Order
    - Page Compaction (more data packed via pages)
      - Stream Merging: None
      - Broadcasting: Major Benefit

Extension

- In the end…
  - Streams: Mostly shorter than full
    - Overall: earlier pages downloaded more often than later pages ← Broadcasting!
  - True Advantages of Stream Merging over Broadcasting?
    - Zero Delay
    - No Batching

Conclusion

- Stream Merging
  - Good when ZERO Delay absolutely necessary.
- In Reality…
  - Can’t have infinite number of streams (channels).
  - Channel Allocation/Scheduling Algorithm:
    - On-line (*i.e.* number of arrivals *unknown*).
    - Optimal Number of Channels : ***O***(N*log*(*p*L)).

References

- Comparison of stream merging algorithms for media-on-demand

  Amoz Bar-Noy, Justin Goshi, Richard E. Ladner, Kenneth Tam

  AT & T Research

  Department of Computer Science and Engineering

  University of Washington

- Efficient Algorithm for Optimal Stream Merging

  Amoz Bar-Noy, Richard E. Ladner

  AT & T Research

  Department of Computer Science and Engineering

  University of Washington

References

- Competitive On-Line Stream Merging Algorithms for Media-on-Demand

  Amoz Bar-Noy, Richard E. Ladner

  AT & T Research

  Department of Computer Science and Engineering

  University of Washington

- The Dyadic Stream Merging Algorithm

  E.G. Coffman, Jr., Predrag Jelenkovic, Petar Momcilovic

  Department of Electrical Engineering

  Columbia University, New York, NY 10027

References

- Provably Efficient Stream Merging

  E.G. Coffman, Jr., Predrag Jelenkovic, Petar Momcilovic

  Department of Electrical Engineering

  Columbia University, New York, NY 10027

- Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems

  Kien A. Hua, Simon Sheu

  Department of Computer Science

  University of Central Florida

  Orlando, FL 32816-2362

Questions?

Thank you!

APPENDIX

Fibonacci Tree Algorithm

- Upper Bound on Optimal Merge Cost

$$M(i,j) = min\{M(i,k\text{-}1) + M(k,j) + (2\mathrm{t}_j\text{-}\mathrm{t}_k\text{-}\mathrm{t}_i)\}$$

$$M(i,j) \leq c(t_j\text{-}t_k)log_2(j\text{-}i+1),\ c = 4log_2 e$$

- Optimal Full Cost for Forest F: *Fcost*(F)
  - $Fcost(\mathrm{F}) = sL + \Sigma_{i=1..s} M(m_i)$
  - s = number of L-trees, $1 \leq i \leq s$
  - m_i = cardinality of each L-tree, $\Sigma_{i=1..s}\, m_i = n$

APPENDIX
Fibonacci Tree Algorithm

- *Fcost*(F)
  - $Fcost(\mathrm{F}) = sL + \Sigma_{i=1..s} M(m_i)$

    $\leq sL + cL\Sigma_{i=1..s} \log_e m_i$
  - By convexity of function $\log_e$:

    $Fcost(\mathrm{F}) \leq sL + cL\Sigma_{i=1..s} \log_e(n/s)$

    $= sL + csL\log_e(n/s)$

    $= sL(c\log_e(n/s) + 1)$

APPENDIX

Fibonacci Tree Algorithm

- Natural Upper Bound on s
  - $sL(c\log_e(n/s) + 1)$ = concave as a function of s
  - Global Maximum of $ce^{-1+1/c}nL$ at $s = e^{-1+1/c}n$

  $\rightarrow s \leq 4N / L$

- $Fcost(\mathrm{F}) \leq 4N / L(c\log_e(n/(4N/L)) + 1)$

  $\in O(N\log(\rho L))$

APPENDIX

Fibonacci Tree Algorithm

- Static Fibonacci Tree Example: L = 26, 13 arrivals…