UNIVERSITY OF OTTAWA
95.573: ALGORITHM ANALYSIS AND DESIGN
FALL 2001

BUBBLES: ADAPTIVE ROUTING SCHEME FOR HIGH-SPEED DYNAMIC NETWORKS

SIAM J. COMPUT.
SHLOMI DOLEV, EVANAKIS KRANAKIS, DANNY KRIZANC, AND DAVID PELEG

SONG GUO

AGENDA

- Introduction
- Routing Schemes
- Bubble Partition
- Bubble Routing
- Edge-Bubble Partition
- Conclusion

INTRODUCTION

- MOTIVATION
  - Improvement of capacities of distributed networks.
  - High-speed networks use a fast switching subsystem.
  - Routing schemes dominate the performance of the distributed network as a whole.

- OBJECTIVE *To minimize the effect of a topology change as much as possible.*

ROUTING SCHEMES: Metrics

- SUPER-HOP COUNT *The super-hop count of a routing scheme bounds the maximum number of super-hops (or, packet generation and transmission steps) required in order to send a message from one node control unit to another.*

- MEMORY *The memory complexity is the total number of bits maintained by the node control units for the routing data-structure.*

- ADAPTABILITY *The adaptability is the maximum number of processors that have to change their portion of the routing distributed database upon a single topology change.*

ROUTING SCHEMES: Basic Routing Schemes (1)

- MULTIPLE SPANNING TREES *The routing distributed database is a description of a spanning tree of the entire topology at each processor.*

- THEOREM *The multiple trees routing scheme has the following properties:*

  1. *super-hop count = 1,*
  2. *memory requirement* $= n^2(\log n + \log \delta)$,
  3. *adaptability = n.*

ROUTING SCHEMES: Basic Routing Schemes (2)

- SINGLE LEADER SCHEME *Only a single processor, L, has a spanning tree of the entire topology. very other processor, P, has the path description to L, path L. Thus, path L is a list of link labels that defines a path from P to L.*

- THEOREM *The single leader scheme has the following properties:*

  1. *super-hop count = 2,*
  2. *memory requirement* = $n(\log n + \log \delta) + n(n-1)\delta$,
  3. *adaptability = n.*

RUBBLE PARTITION: Basic Concepts (1)

- **[a, b]-BUBBLE PARTITION** *Given a graph G and parameter 0<a<b, an [a, b]-bubble partition of G is a partition of the nodes of G into disjoint connected components called bubbles, $P=\{B_1,\ldots,B_r\}$, such that the size of each bubble B satisfies $a \leqslant |B| \leqslant b$.*

- **THEOREM** *For every n-node connected graph G with maximum degree δ and parameter $1 \leqslant x \leqslant n$, G has an $[x, \delta x]$-bubble partition.*

BUBBLE PARTITION: Basic Concepts (2)

- **ALGORITHM PARTITION(*T*, *x*)**

(1) **While** $\mathcal{T}$ contains more than δx nodes **do:**
  (a) Mark every node P in $\mathcal{T}$ by M_P, the total number of nodes in the subtree $\mathcal{T}_P$ rooted at P.
  (b) Find a node P with the property that $M_P \geq x$, but all of its children Q satisfy $M_Q < x$.
  (c) Make $\mathcal{T}_P$ into a bubble.
  (d) Remove this bubble and the edge connecting P to its parent from $\mathcal{T}$.
  End_while
(2) Make $\mathcal{T}$ into a bubble and terminate.

BUBBLE PARTITION: Hierarchical Partition (1)

- **PARTITION REFINEMENT** *Given two partitions P and P', P is a refinement of P' if each bubble of P is fully contained in some bubble of P'.*

- **$\bar{x}$ -HIERARCHICAL BUBBLE PARTITION** *Given a graph G and a list of k integer parameters* $\bar{x} = (x_1, x_2, \ldots, x_k, x_{k+1})$, *where* $x_1 = n$, $x_i > \delta x_{i+1}$, *and* $x_{k+1} = 1$, *an* $\bar{x}$ *-hierarchical bubble partition of G is a collection of k partitions* P_i, $1 \leqslant i \leqslant k$, *with the following properties:*

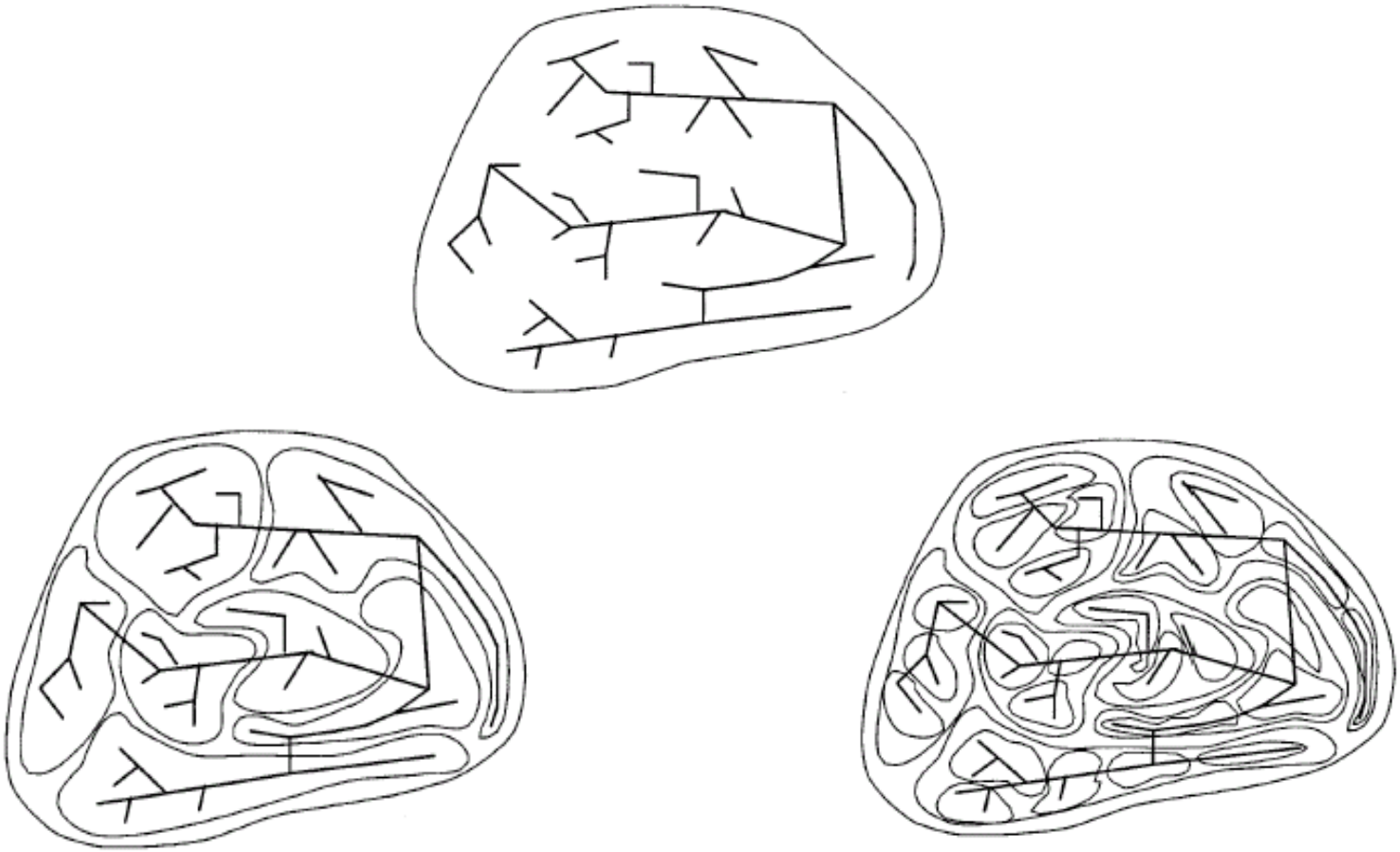
  1. *For every* $1 \leqslant i \leqslant k$, P_i *is an* $[x_i, \delta x_i]$*-bubble partition.*
  2. *For every* $1 \leqslant i \leqslant k$, P_i *is a refinement of* P_{i-1}.

BUBBLE PARTITION: Hierarchical Partition (2)

- **ALGORITHM HIERARCHY($\bar{x}$)**

(1) Choose an arbitrary node of the graph, R.
(2) Construct a spanning tree $\mathcal{T}_R$ rooted at R with edges directed towards the leaves.
(3) Let the first level partition $\mathcal{P}_1$ consist of a single bubble covering the entire graph.
(4) **For** $i = 2$ to k **do:**
For every bubble $\mathcal{B}$ of partition $\mathcal{P}_{i-1}$ **do:**
(a) Let $\mathcal{T}(\mathcal{B})$ be the portion of $\mathcal{T}_R$ that spans the bubble $\mathcal{B}$.
(b) Apply Algorithm PARTITION($\mathcal{T}(\mathcal{B}), x_i$) to $\mathcal{T}(\mathcal{B})$.
(c) Add the resulting bubbles to the current level partition $\mathcal{P}_i$.

BUBBLE PARTITION: Hierarchical Partition (3)



BUBBLE PARTITION: Hierarchical Partition (4)

PROPERTIES

1. *We choose the parameters* $\bar{x}$ *as* $x_i = \left\lfloor n^{(k-i+1)/k} \right\rfloor$, *which satisfies the requirement of the definition as long as* $n^{1/k} \geq \delta$.

2. $F_{i+1} \subseteq F_i$ *for every* $1 \leqslant i \leqslant k-1$, *where* $F_i = \bigcup_{B \in P_i} T(B)$

THEOREM *Every graph has an* $\bar{x}$ *-hierarchical bubble partition.*

BUBBLE PARTITION: Basic Operations

PROCEDURE COMBINE(B_l',B_l'')

Input: Two tree-neighboring level l bubbles $\mathcal{B}_l'$ and $\mathcal{B}_l''$.

Action:
Connect $\mathcal{T}(\mathcal{B}_l')$ and $\mathcal{T}(\mathcal{B}_l'')$ using their connecting link.

Output: A combined bubble $\mathcal{B}_l$.

PROCEDURE SPLIT(B_l)

Input: A (typically oversized) bubble $\mathcal{B}_l$.

Action:
Select one of the nodes of $\mathcal{B}_l$ to be the root R.
Direct $\mathcal{T}(\mathcal{B}_l)$ from R towards the leaves.
Execute Algorithm PARTITION$(\mathcal{T}(\mathcal{B}_l), x_l)$.

Output: An $[x_l, \delta x_l]$-bubbles partition of $\mathcal{B}_l$.

BUBBLE PARTITION: Procedure REMOVE (1)

PROCEDURE REMOVE(*e, Fl, Fl-1*)

Input: Edge e for removal, forests $\mathcal{F}_l$ and $\mathcal{F}_{l-1}$ spanning legal partitions.

Action:

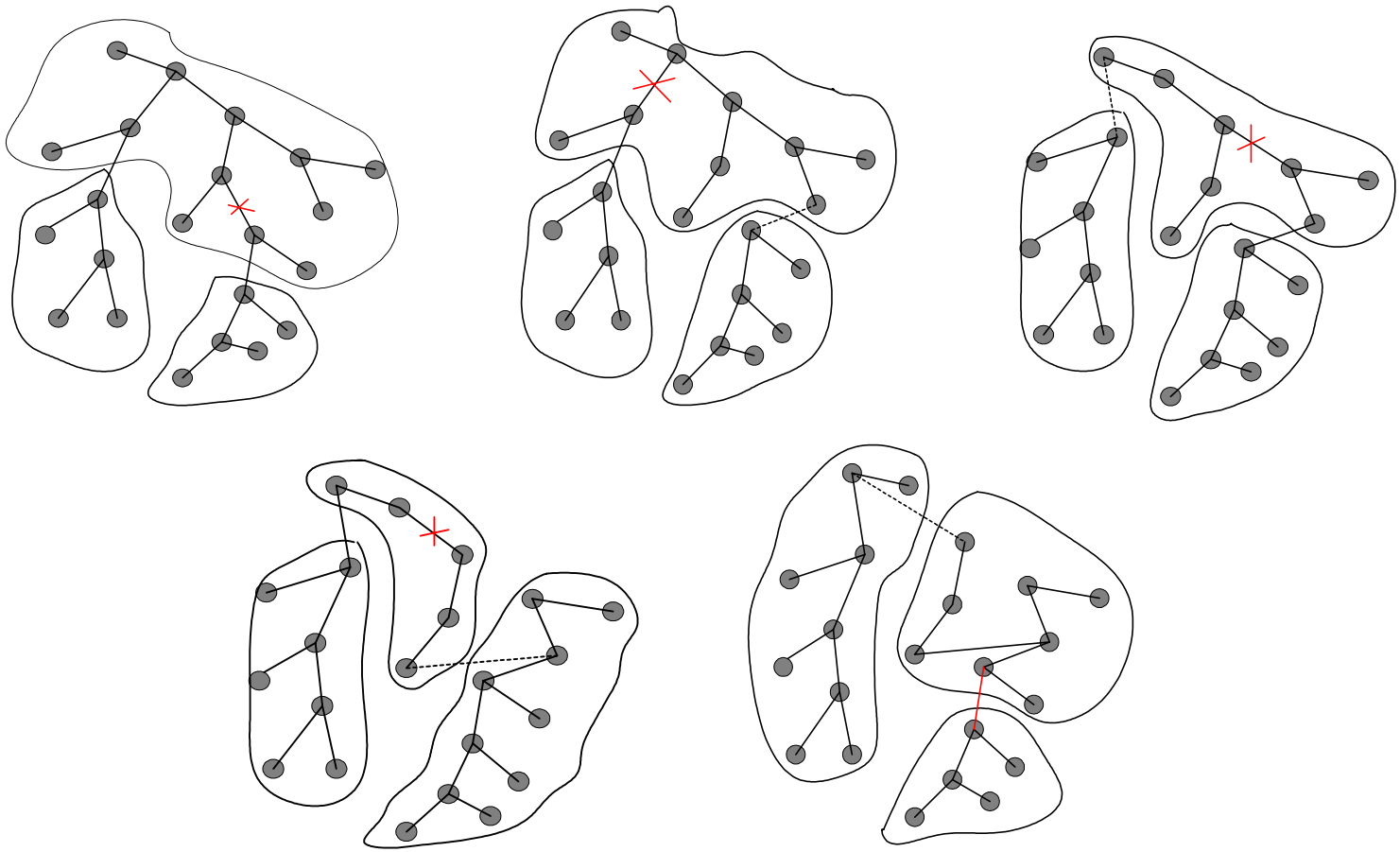
The removal of e causes the partition of a single bubble, say $\mathcal{B}_l$, into two portions, denoted $\mathcal{B}'_l$ and $\mathcal{B}''_l$.

For each of these two portions $\hat{\mathcal{B}}$ **do:**

1. If the size of $\hat{\mathcal{B}}$ is still greater than x_l, then make it into an independent bubble without change.
2. Otherwise, if it is too small, then do the following:
   (a) Find some bubble $\tilde{\mathcal{B}}$ of $\mathcal{F}_l$ that is a tree-neighbor of $\hat{\mathcal{B}}$ w.r.t. $\mathcal{F}_{l-1}$.
   (b) Merge $\tilde{\mathcal{B}}$ and $\hat{\mathcal{B}}$ into $\bar{\mathcal{B}}$ by invoking procedure COMBINE$(\tilde{\mathcal{B}}, \hat{\mathcal{B}})$.
   (c) If $\bar{\mathcal{B}}$ includes more than δx_l nodes, then split it by invoking procedure SPLIT$(\bar{\mathcal{B}})$.

Output: Forest $\mathcal{F}'_l$.

BUBBLE PARTITION: Procedure REMOVE (2)



BUBBLE PARTITION: Procedure REMOVE (3)

THEOREM

1. *If F_l is the spanning forest of a legal $[x_l, \delta x_l]$-bubble partition, then after the execution of Procedure* REMOVE*(e, F_l, F_{l-1}) and removing the edge e, the output forest F_l' represents a legal $[x_l, \delta x_l]$-bubble partition as well.*

2. *Every edge added to F_l' during the execution of* REMOVE*(e, F_l, F_{l-1}) is an edge of F_{l-1}.*

3. *The number of edges removed from F_l during the execution of* REMOVE*(e, F_l, F_{l-1}) is at most 3.*

BUBBLE PARTITION: Procedure REMOVEALL (1)

- **PROCEDURE REMOVEALL(*e, i*)**

Input: Edge e for removal,
forests $\mathcal{F}_l$ for $i-1 \leq l \leq k$ spanning legal partitions.

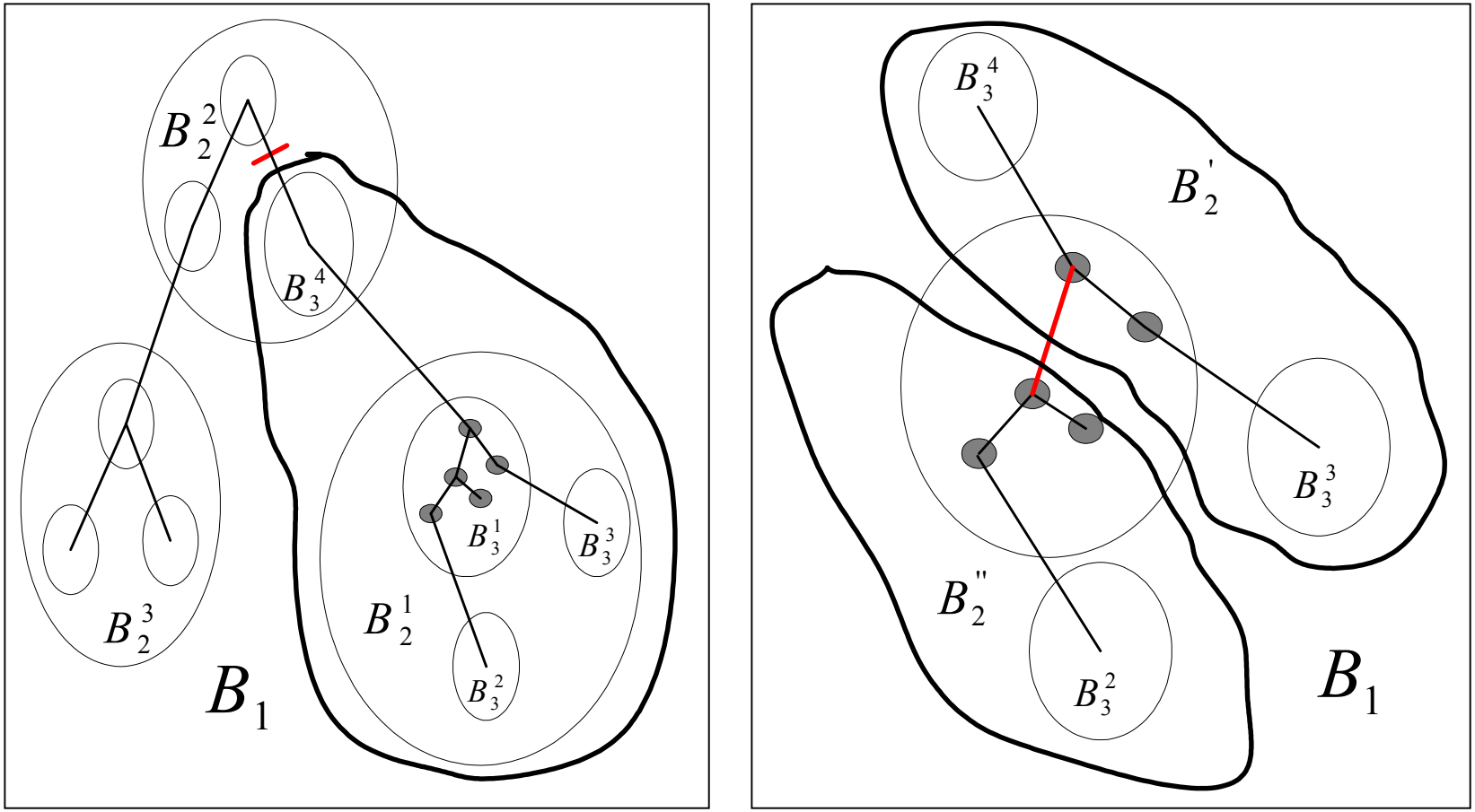
Action:

Let $\mathcal{F}'_i \leftarrow \text{REMOVE}(e, \mathcal{F}_i, \mathcal{F}_{i-1})$.
For $l = i+1$ to k **do:**

1. Let $\{e_1, e_2, \ldots, e_m\} = \mathcal{F}_l - \mathcal{F}'_{l-1}$.
2. Let $\mathcal{F}_l^{(0)} \leftarrow \mathcal{F}_l$.
3. **For** $j = 1$ to m **do:**
   If e_j is in $\mathcal{F}_l^{(j-1)}$, then invoke $\mathcal{F}_l^{(j)} \leftarrow \text{REMOVE}(e_j, \mathcal{F}_l^{(j-1)}, \mathcal{F}_{l-1})$.
4. Let $\mathcal{F}'_l \leftarrow \mathcal{F}_l^{(m)}$.

Output: Forests $\mathcal{F}'_l$ for $i \leq l \leq k$.

BUBBLE PARTITION: Procedure REMOVEALL (2)



BUBBLE PARTITION: Procedure REMOVEALL (3)

THEOREM

1. *If F_l is the spanning forest of a legal $[x_l, \delta x_l]$-bubble partition for every $i\text{-}1 \leqslant l \leqslant k$, then after the execution of Procedure* REMOVEALL*(e, i) and removing the edge e, the output forest F_l' represents a legal $[x_l, \delta x_l]$-bubble partition as well for every $i \leqslant l \leqslant k$.*
2. *The output partition P_l' is a refinement of P_{l-1}' (i.e., $F_{i+1} \subseteq F_i$) for every $i \leqslant l \leqslant k$.*
3. *The number of edges removed from F_l during the execution of* REMOVEALL*(e, F_l, F_{l-1}) is 3^{l-i+1} at most for every $i \leqslant l \leqslant k$.*

BUBBLE PARTITION: Dynamic Maintenance (1)

LINK REMOVAL

(1) If the removal of the link e does not partition the spanning tree of any bubble, then no changes of the hierarchical bubble partition is required.

(2) Otherwise the link removal is handled at each level starting from level 1 and moving up.

(a) At level 1, if the communication graph is still connected, then a non-tree link is promoted to be a tree link in order to retain the connectivity of the entire spanning tree.

(b) The removal itself is then performed by applying Procedure REMOVEALL*(e,2,).*

BUBBLE PARTITION: Dynamic Maintenance (2)

LINK ADDITION

(1) If an addition of the link does not connect two previously separated connected components (G' and G"), then no changes of the routing database is required.

(2) Otherwise, let $T(B_1')$ and $T(B_1'')$ be the spanning tree of G' and G" before the link addition.

(a) Connect $T(B_1')$ with $T(B_1'')$ by invoking COMBINE*(B_1', B_1'').*

(b) Continue with level 2 up to k. If the new link connects two legal bubbles, then no update operations are triggered by level i. Otherwise combine the illegal bubble with the new tree-neighboring bubble (using COMBINE*) and split the combined bubble if necessary (using* SPLIT *and* REMOVEALL*).*

BUBBLE PARTITION: Dynamic Maintenance (3)

- **NODE ADDITION**

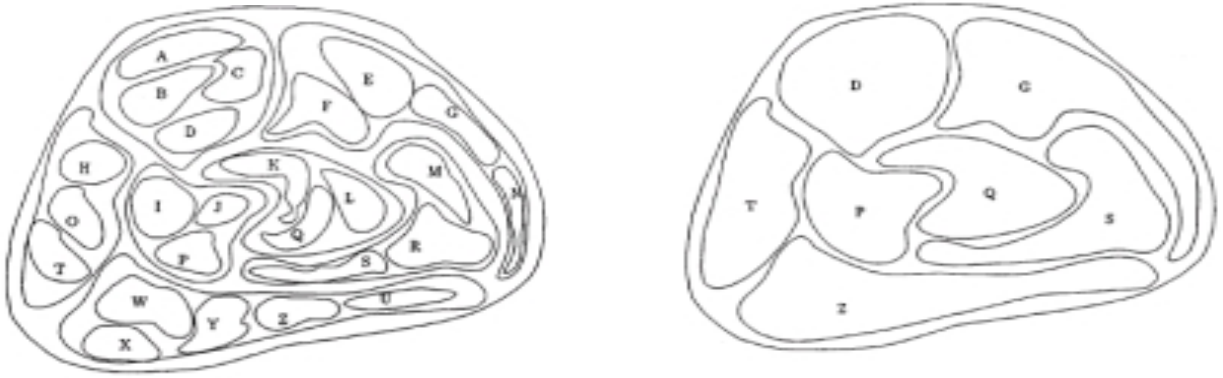
  The addition is handled by first adding a link that connects two separated connected components one of which is the single node. Then adding the rest of the links one by one.

- **NODE REMOVAL**

  Node removal is done by removing one link at a time and then removing the node.

BUBBLE ROUTING: Distributed Database

- **BUBBLE MEMBER** *At level k, define the node that reside in B_k as its members, and choose one of them to be the leader of the bubble; for i<k, define the members of a bubble B_i to be all the leaders of level i+1 bubbles that reside in B_i, then choose one of them to be the leader of B_i.*



- **DATABASE** *The Tree $T(B_i)$ is known to every member of B_i.*

BUBBLE ROUTING: Routing Strategy

- **ROUTE DISCOVERY** *Request for routes from its leader recursively, and must terminate at a member of* B_1 *in the worst case.*

- **ROUTE MAINTENANCE** *Whenever we apply the Procedure* COMBINE and *Procedure* SPLIT, *the members of new created bubbles should be updated.*

BUBBLE ROUTING: Performance

- **PROPERTIES**

  1. *The number of members in a bubble of level i is at most* $\pi i = \delta xi/xi_{+1}$.

  2. *The total number of members in all level i bubbles is at most* n/x_{i+1}.

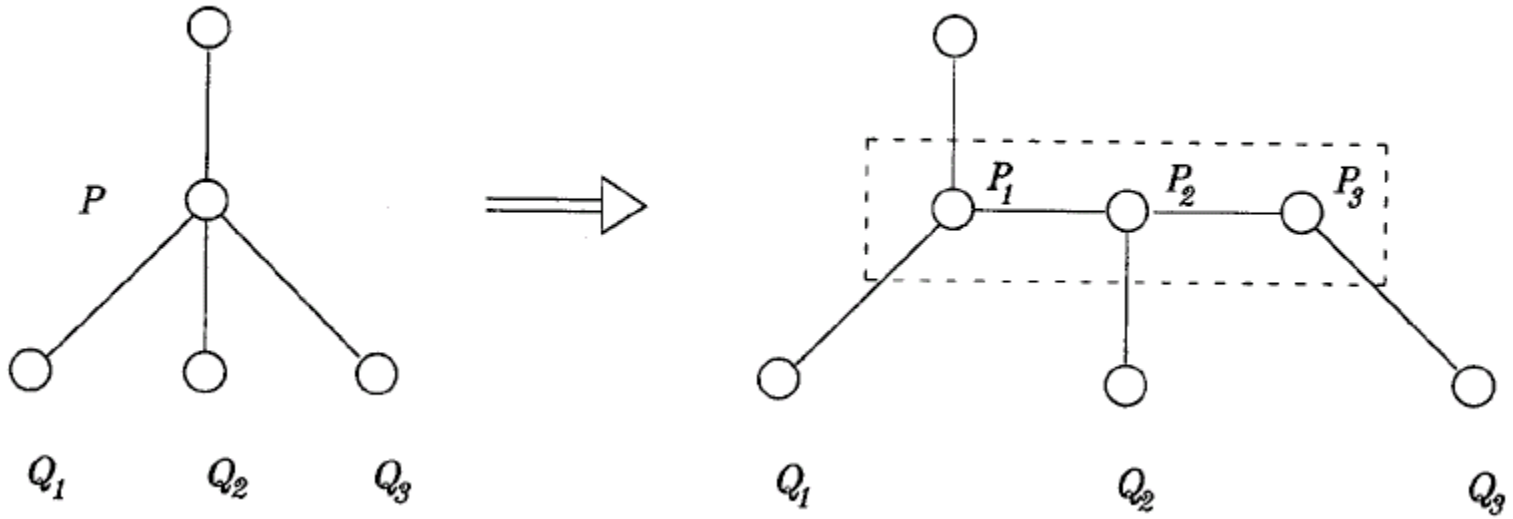
- **THEOREM** *The bubble routing scheme has the following properties if* $x_i = \left| n^{(k-i+1)/k} \right|$ *for* $1 \leqslant i \leqslant k$:

  1. *Super-hop count* $= k$

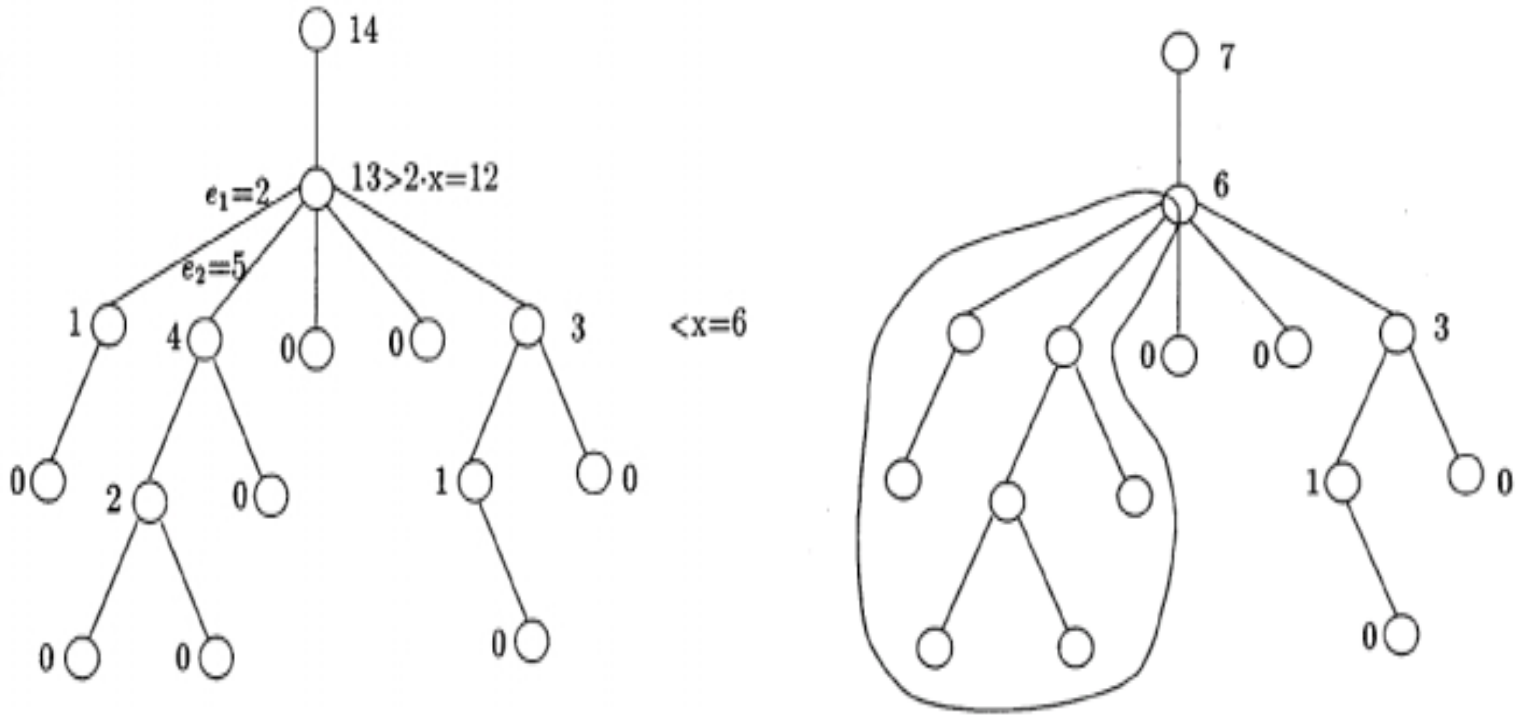
  2. *Memory requirement* $= n(\log n + \log\delta)\sum_{1 \leqslant i \leqslant k} \pi i = k\delta\ (\log n + \log\delta)\ n^{1+1/k}$

  3. *Adaptability* $= \delta \sum_{1 \leq i \leq k} 3^{i-1}\pi_i = 3^k \delta^2 n^{1/k}$ *in the node-failure model and* $\sum_{1 \leq i \leq k} 3^{i-1}\pi_i = 3^k \delta n^{1/k}$ *in the link-failure model.*

EDGE PARTITION : Intuitive Approach



EDGE PARTITION : Partition Algorithm



CONCLUSION

	Super-hop count	Memory required	Adaptability	Model
Multiple trees	1	$O(n^2 \log n)$	n	
Single leader	2	$O(n^2 \log \delta)$	n	
Basic bubbles	k	$O(k\delta n^{1+1/k} \log \delta)$	$O(3^k \delta^2 n^{1/k})$	Node failures
Basic bubbles	k	$O(k\delta n^{1+1/k} \log \delta)$	$O(3^k \delta n^{1/k})$	Link failures
Edge-bubbles	k	$O(kn^{1+1/k} \log \delta)$	$O(3^k \delta n^{1/k})$	Node failures
Edge-bubbles	k	$O(kn^{1+1/k} \log \delta)$	$O(3^k n^{1/k})$	Link failures