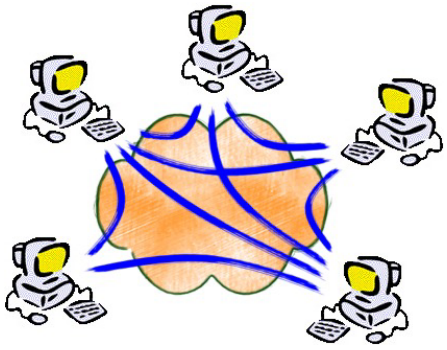
# *Protocols for Large Self-Organizing Peer Networks*

*Jorg Liebeherr*

*University of Virginia*

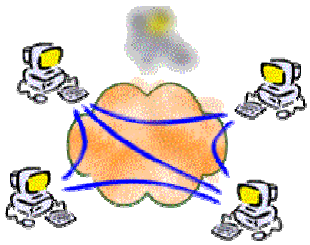




## HyperCast Project

- **HyperCast** is a set of protocols for large-scale overlay multicasting and peer-to-peer networking

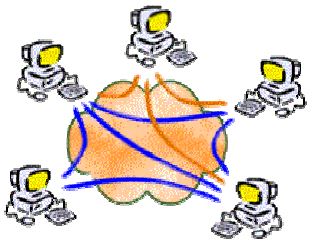
- **Motivating Research Problems:**
  - How to organize thousands of applications in a virtual overlay network?
  - How to do multicasting in very large overlay networks?



# Acknowledgements

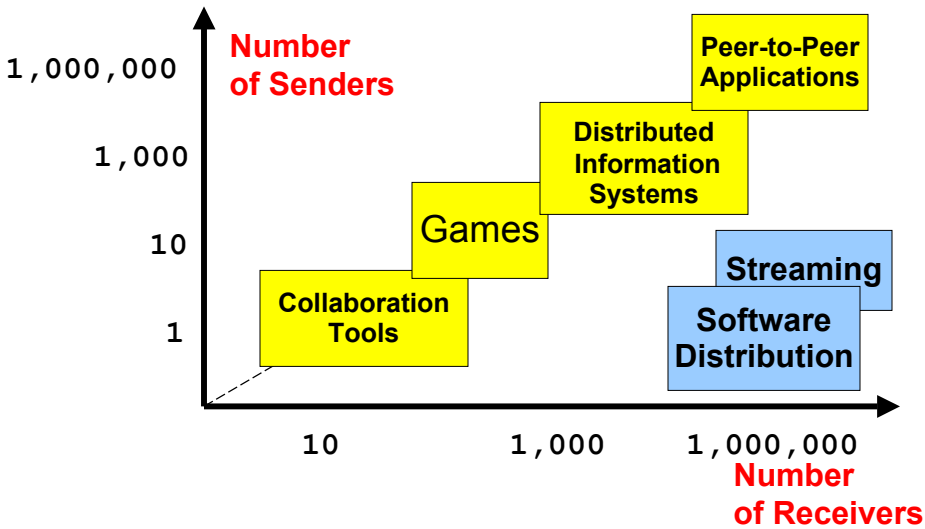

- **Team**:
  - *Past: Bhupinder Sethi, Tyler Beam, Burton Filstrup, Mike Nahas, Dongwen Wang, Konrad Lorincz, Jean Ablutz*
  - *Current:*, Weisheng Si, Haiyong Wang, Jianping Wang, Guimin Zhang
- This work is supported in part by the National Science Foundation:

D E N A L I

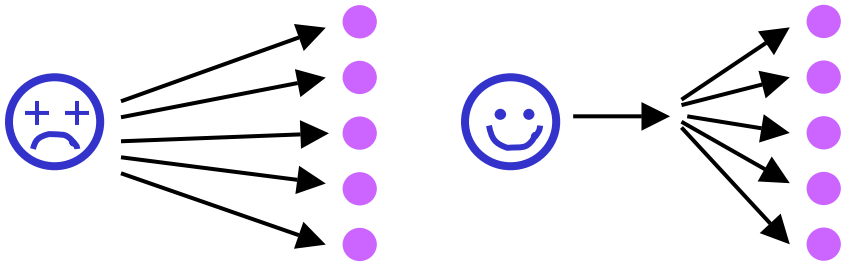





# Applications with many receivers



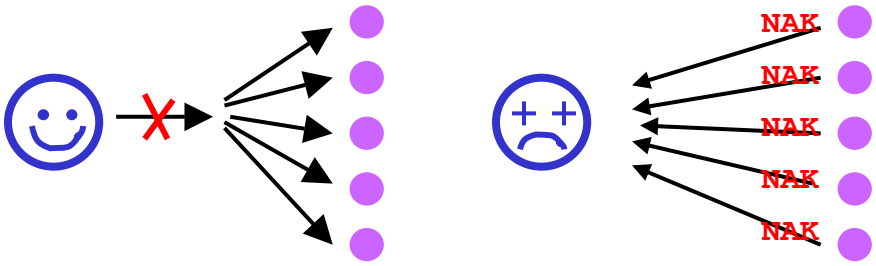
## Need for Multicasting ?



- Maintaining unicast connections is not feasible
- Infrastructure or services needs to support a *"send to group"*

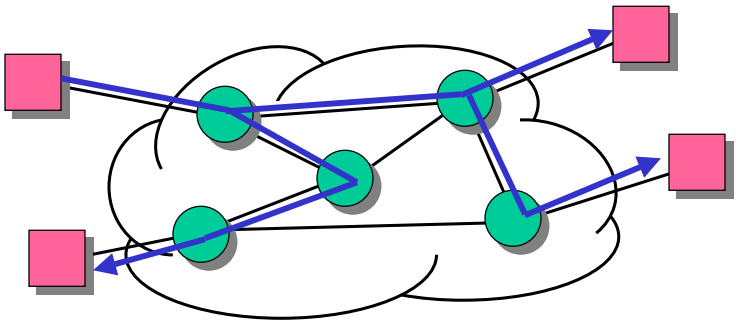


## Problem with Multicasting




- **Feedback Implosion:** A node is overwhelmed with traffic or state
  - One-to-many multicast with feedback (e.g., reliable multicast)
  - Many-to-one multicast (Incast)

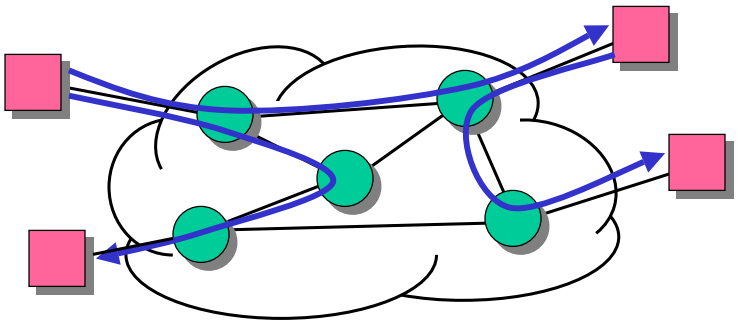
## Multicast support in the network infrastructure (IP Multicast)



- **Reality Check** (after 10 years of IP Multicast):
  - Deployment has encountered severe scalability limitations in both the size and number of groups that can be supported
  - IP Multicast is still plagued with concerns pertaining to scalability, network management, deployment and support for error, flow and congestion control



## Overlay Multicasting



- **Logical overlay** resides on top of the Layer-3 network
- Data is transmitted between neighbors in the overlay
- No network support needed
- Overlay topology should match the Layer-3 infrastructure

## Overlay-based approaches for multicasting

- Build an overlay mesh network and embed trees into the mesh:
  - Narada (CMU)
  - RMX/Gossamer (UCB)
  - many more
- Build a shared tree:
  - Yallcast/Yoid (NTT, ACIRI)
  - AMRoute (Telcordia, UMD – College Park)
  - Overcast (MIT)
  - many more
- Build an overlay using a "logical coordinate spaces":
  - Chord (UCB, MIT) ← *not used for multicast*
  - CAN (UCB, ACIRI)

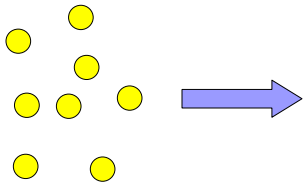


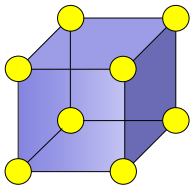
## HyperCast Approach

- **Build overlay network as a graph with known properties**
  - N-dimensional (incomplete) hypercube
  - Delaunay triangulation
- **Advantages:**
  - Achieve good load-balancing
  - Exploit symmetry
  - Routing in the overlay comes for free
- **Claim: Can improve scalability of multicast and peer-to-peer networks by orders of magnitude over existing solutions**

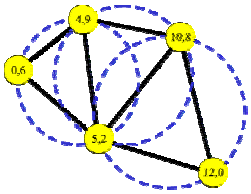
## Hypercast Software

- Applications organize themselves to form a logical overlay network with a given topology
  - No central control
  - Dynamic membership





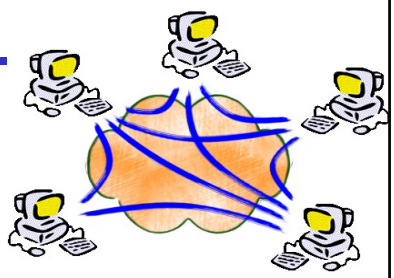
*hypercube*



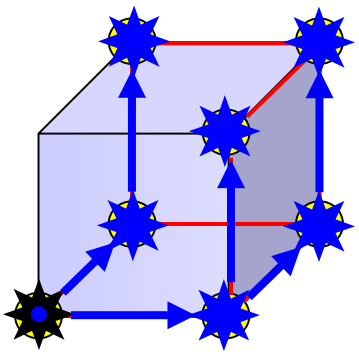

*Delaunay triangulation*



## Data Transfer

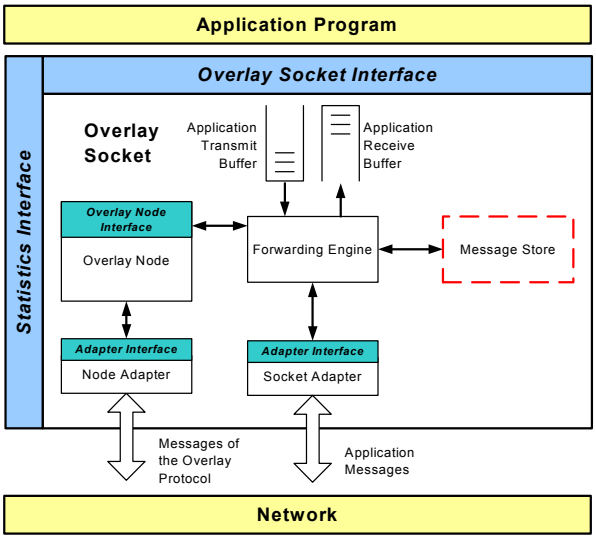


- Data is distributed neighbor-to-neighbor in the overlay network



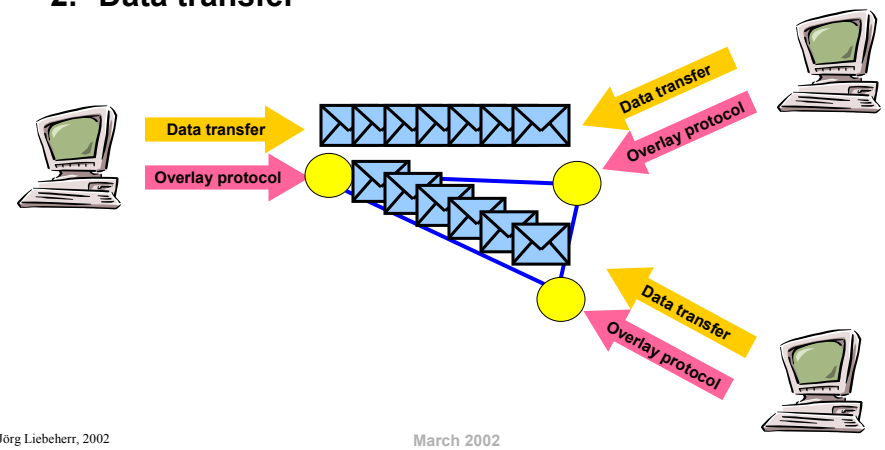
# HyperCast Software: Overlay Socket

- **Transport services in Peer-to-Peer Networks**
- Socket-based API
- UDP or TCP
- Different reliability semantics
- Implementation done in Java
- Software available from: www.cs.virginia.edu/~hypercast

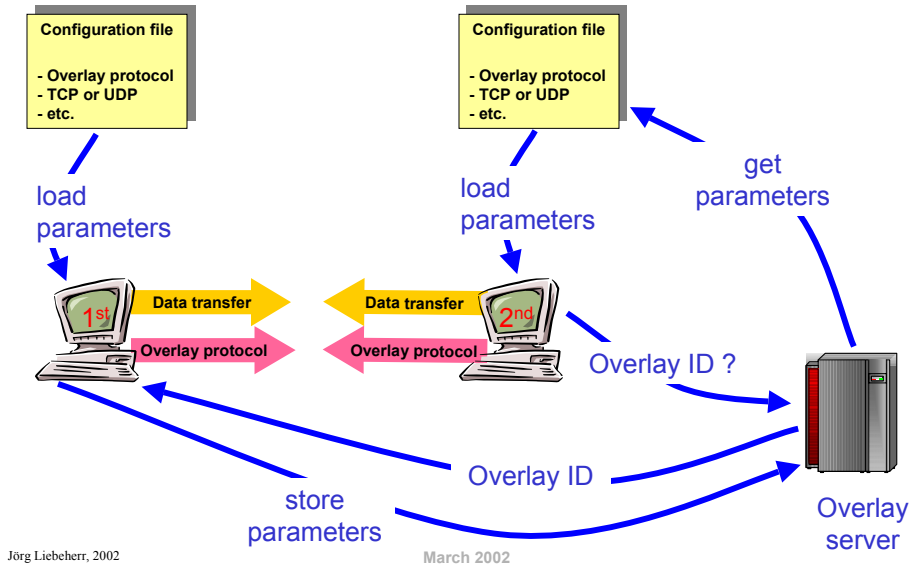



# HyperCast Software: Data Exchange

- **Each overlay socket has two communication ports:**
  1. **Protocol to manage the overlay (overlay protocol)**
  2. **Data transfer**



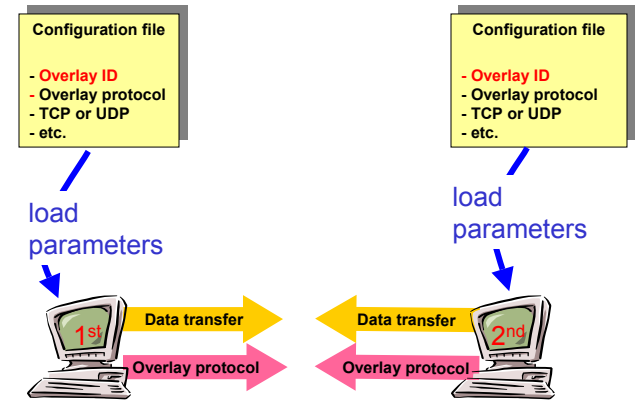
# HyperCast Software: Bootstrap





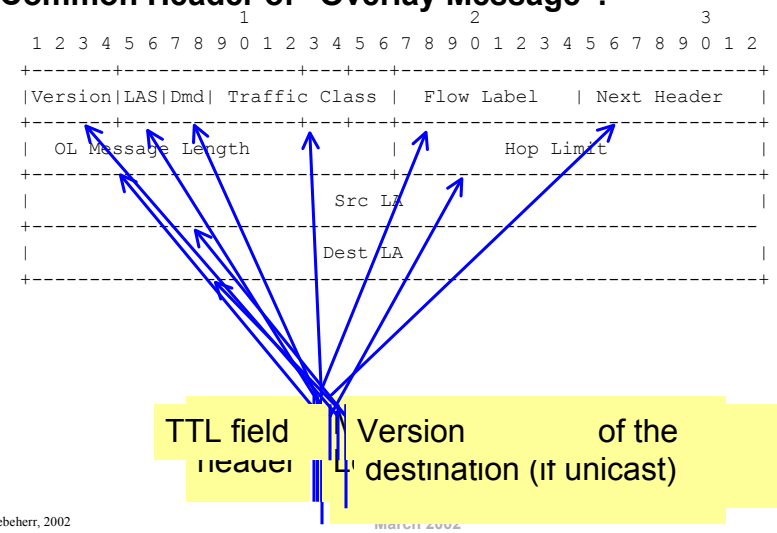

# HyperCast Software: Bootstrap

## (without Overlay server)



# HyperCast Software: Message Formats

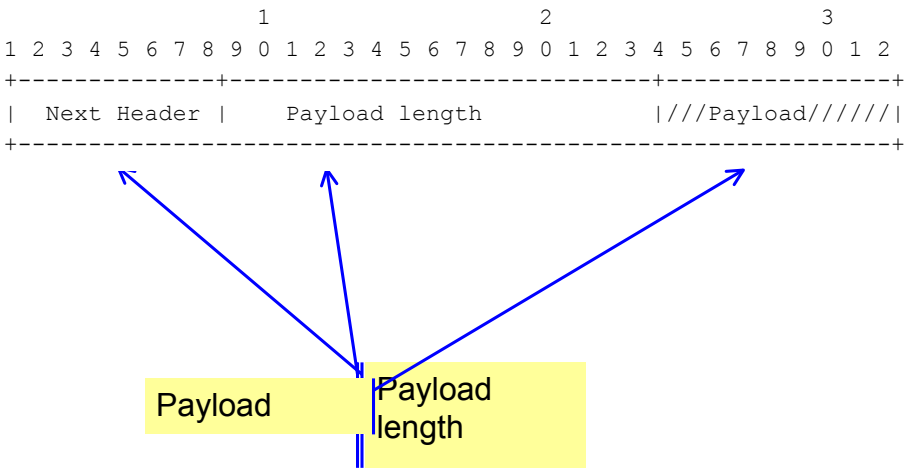
- **Common Header of "Overlay Message":**






# HyperCast Software: Message Formats

- **"Raw" messages**



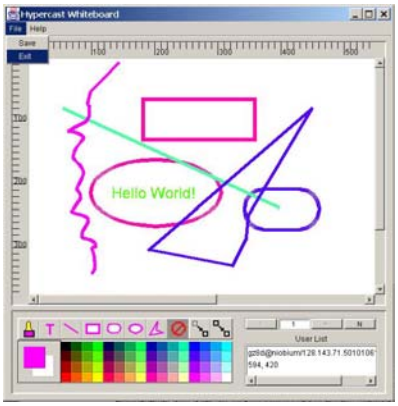
# HyperCast Software: Socket Based API

```
//Generate the configuration object
OverlayManager om = new OverlayManager(propertyfilename);
String overlayID = om.getDefaultProperty("OverlayID")
OverlaySocketConfig config = new
om.getOverlaySocketConfig(overlayID);

//create an overlay socket
OL_Socket socket = config.createOverlaySocket(callback);

//Join an overlay
socket.joinGroup();

//Create a message
OL_Message msg = socket.createMessage(byte[] data, int length);

//Send the message to all members in overlay network
socket.sendToAll(msg);

//Receive a message from the socket
OL_Message msg = socket.receive();

//Extract the payload
byte[] data = msg.getPayload();
```

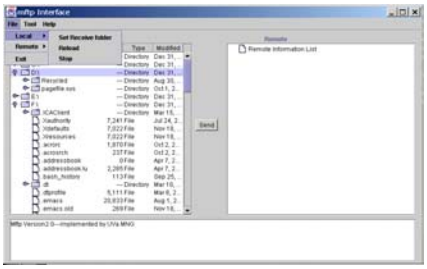


# HyperCast Software: Demo Applications

*Distributed Whiteboard*



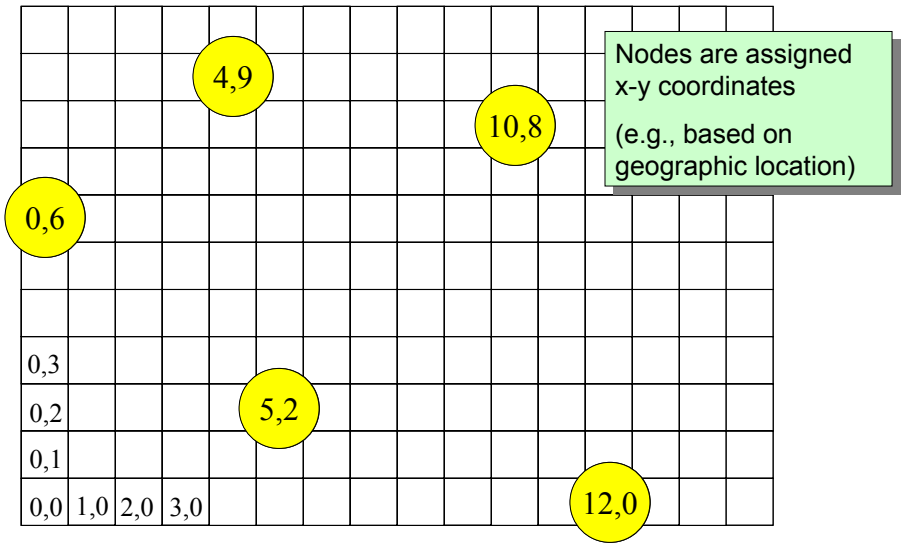

*Multicast file transfer*



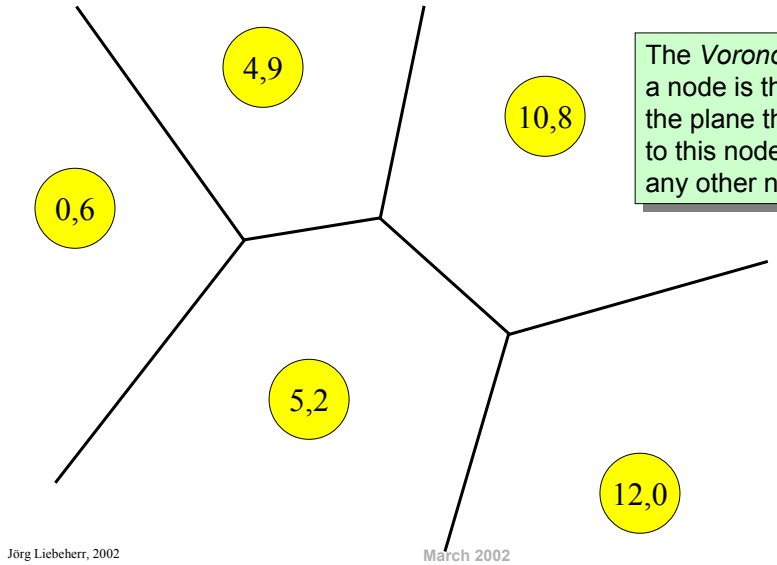
# Delaunay Triangulation Overlays



# Nodes in a Plane



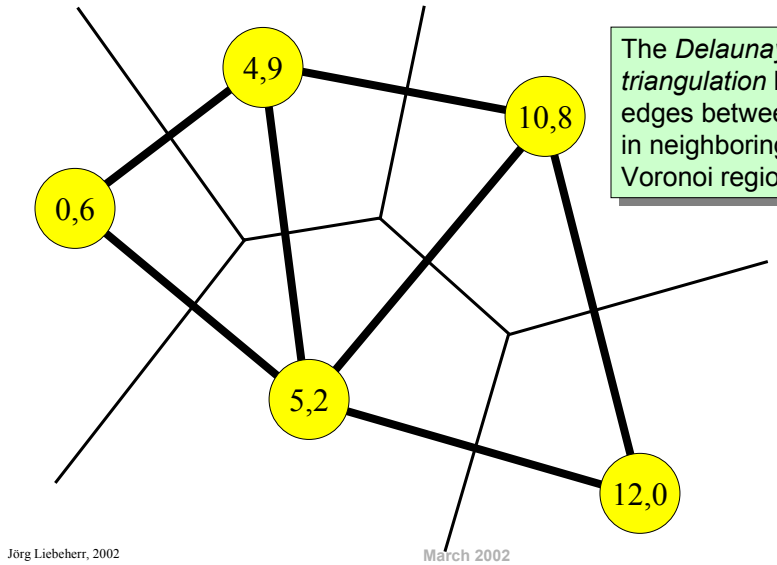
## Voronoi Regions




The *Voronoi region* of a node is the region of the plane that is closer to this node than to any other node.

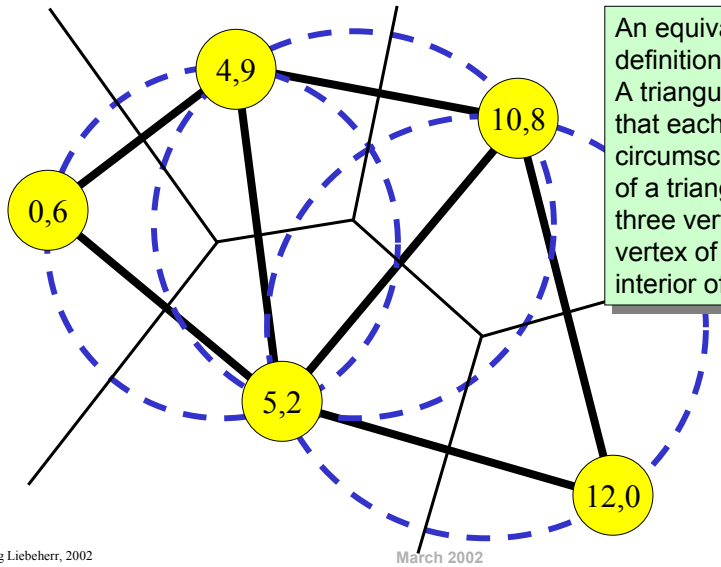


## Delaunay Triangulation




The *Delaunay triangulation* has edges between nodes in neighboring Voronoi regions.

## Delaunay Triangulation



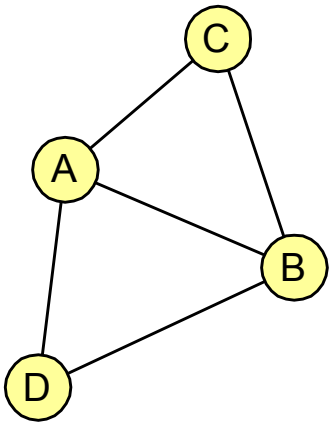

An equivalent definition:
A triangulation such that each circumscribing circle of a triangle formed by three vertices, no vertex of is in the interior of the circle.



## Locally Equiangular Property

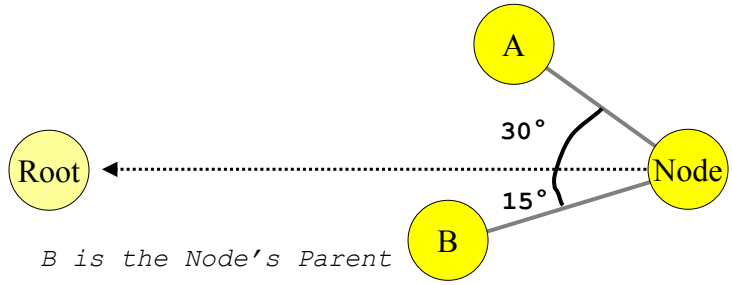
- Sibson 1977: **Maximize the minimum angle**

  For every convex quadrilateral formed by triangles ACB and ABD that share a common edge AB, the minimum internal angle of triangles ACB and ABD is at least as large as the minimum internal angle of triangles ACD and CBD.



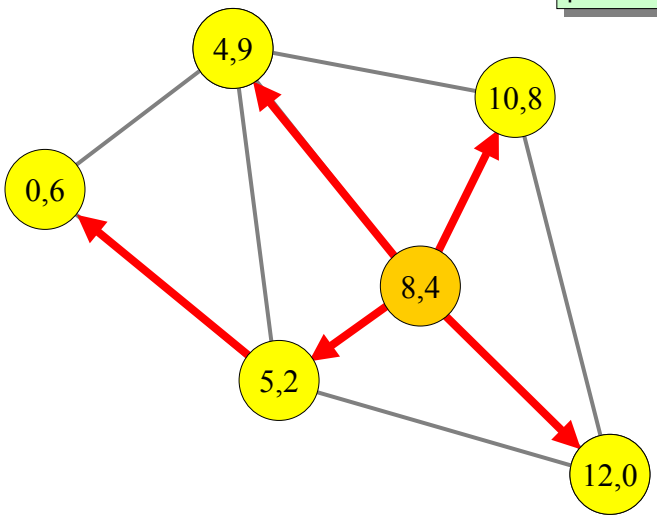
# Next-hop routing with Compass Routing

- A node's parent in a spanning tree is its neighbor which forms the smallest angle with the root.
- A node need only know information on its neighbors – no routing protocol is needed for the overlay.



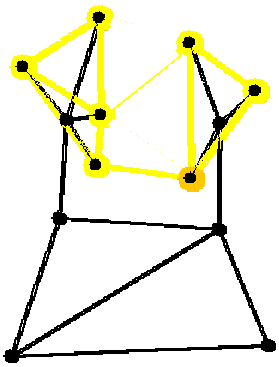



Spanning tree when node (8,4) is root. The tree can be calculated by both parents and children.



## Problem with Delaunay Triangulations

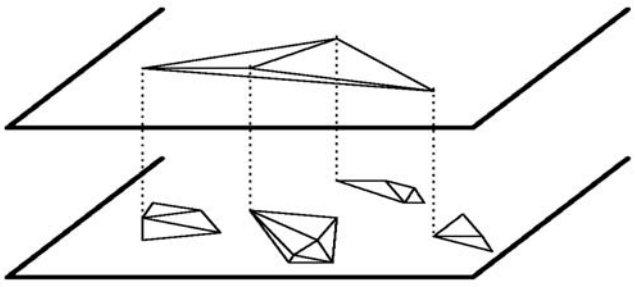
- Delaunay triangulation considers location of nodes, but not the network topology
- 2 heuristics to achieve a better mapping





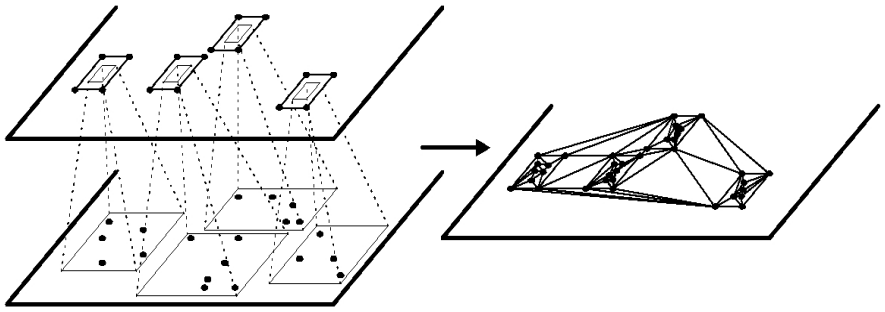
## Hierarchical Delaunay Triangulation

- 2-level hierarchy of Delaunay triangulations
- The node with the lowest x-coordinate in a domain DT is a member in 2 triangulations



## Multipoint Delaunay Triangulation

- Different (“implicit”) hierarchical organization
- “Virtual nodes” are positioned to form a “bounding box” around a cluster of nodes. All traffic to nodes in a cluster goes through one of the virtual nodes

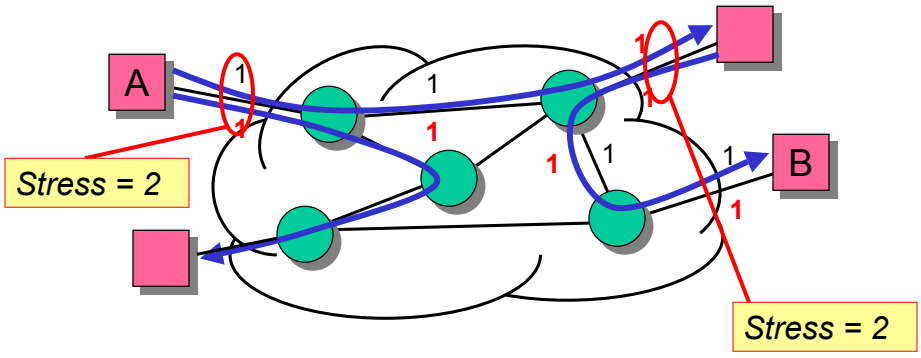




## Evaluation of Overlays

- **Simulation:**
  - Network with 1024 routers (“Transit-Stub” topology)
  - 2 - 512 hosts
- **Performance measures for trees embedded in an overlay network:**
  - Degree of a node in an embedded tree
  - “Relative Delay Penalty”: Ratio of delay in overlay to shortest path delay
  - “Stress”: Number of duplicate transmissions over a physical link

## Illustration of “Stress” and “Relative Delay Penalty”




Unicast delay A→B : 4

Delay A→B in overlay: 6

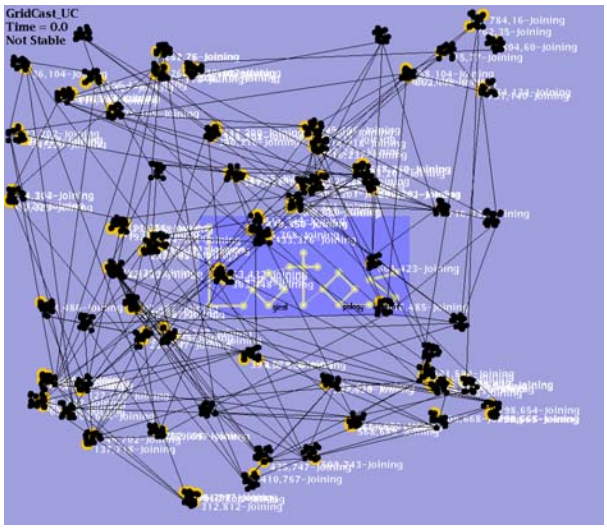
Relative delay penalty for A→B: 1.5



## Transit-Stub Network

Transit-Stub

- GA Tech topology generator
- 4 transit domains
- 4×16 stub domains
- 1024 total routers
- 128 hosts on stub domain



## Overlay Topologies

Delaunay Triangulation and variants
- Hierarchical DT
- Multipoint DT

Degree-6 Graph
- Similar to graphs generated in Narada

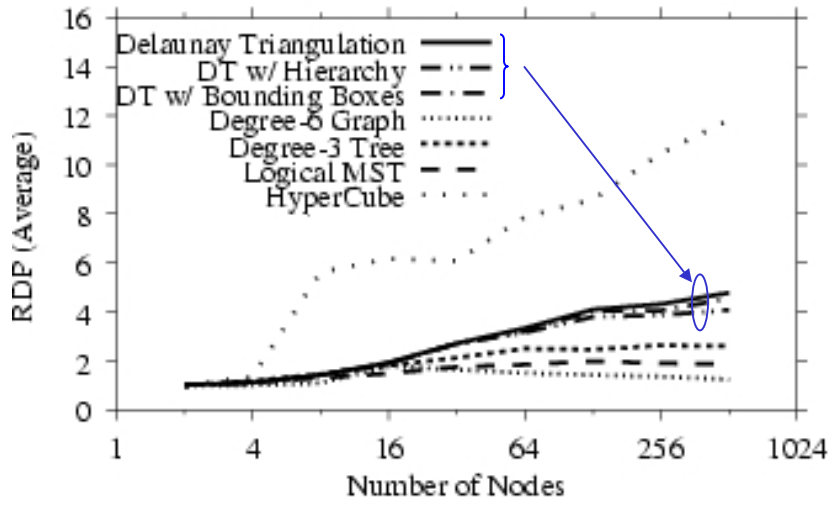
Degree-3 Tree
- Similar to graphs generated in Yoid

Logical MST
- Minimum Spanning Tree

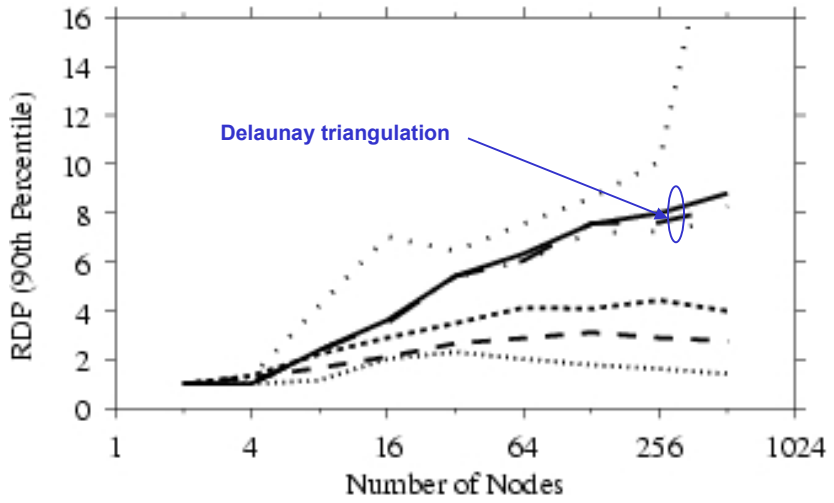
Hypercube



## Average Relative Delay Penalty

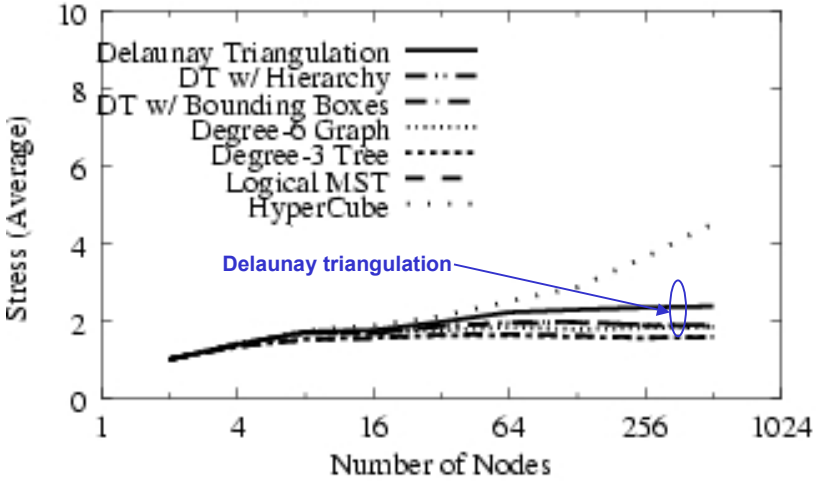


## 90th Percentile of Relative Delay Penalty

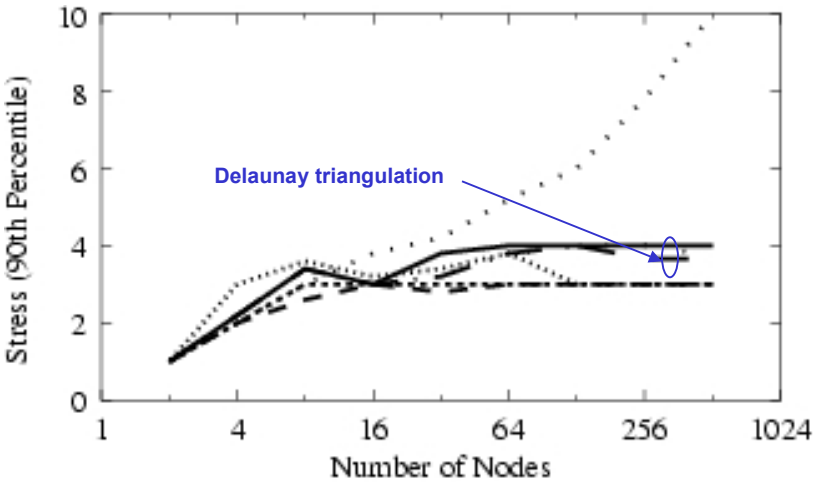





## Average "Stress"



## 90th Percentile of "Stress"





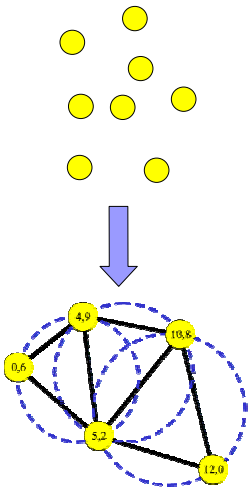

## The DT Protocol

Protocol which organizes members of a network in a Delaunay Triangulation

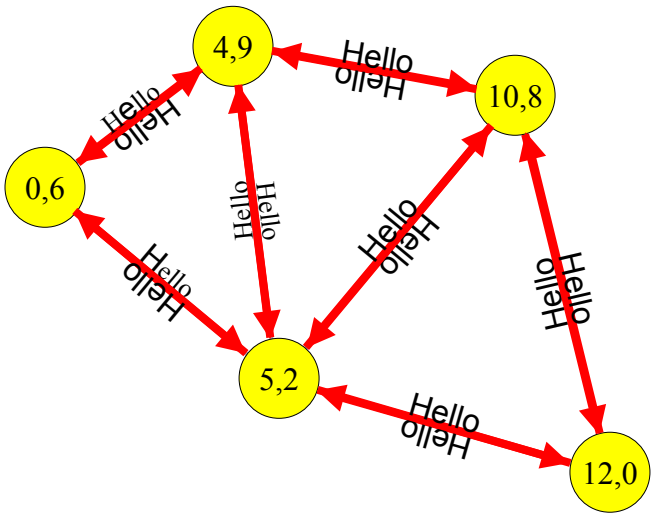
- Each member only knows its neighbors
- "soft-state" protocol

**Topics:**

- Nodes and Neighbors
- Example: A node joins
- State Diagram
- Rendezvous
- Measurement Experiments

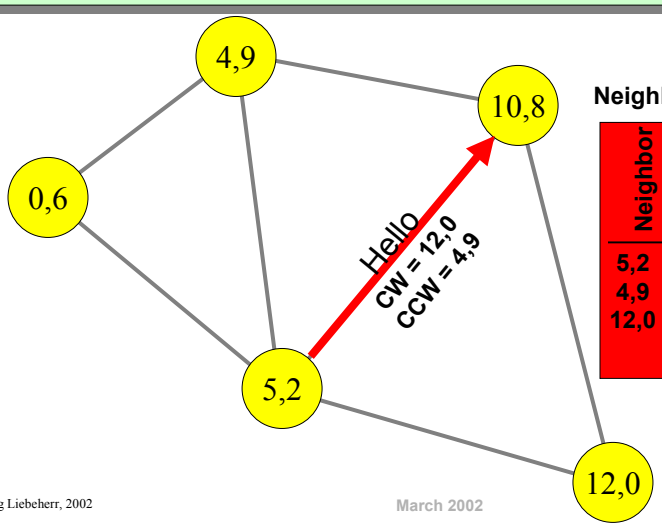


Each node sends Hello messages to its neighbors periodically



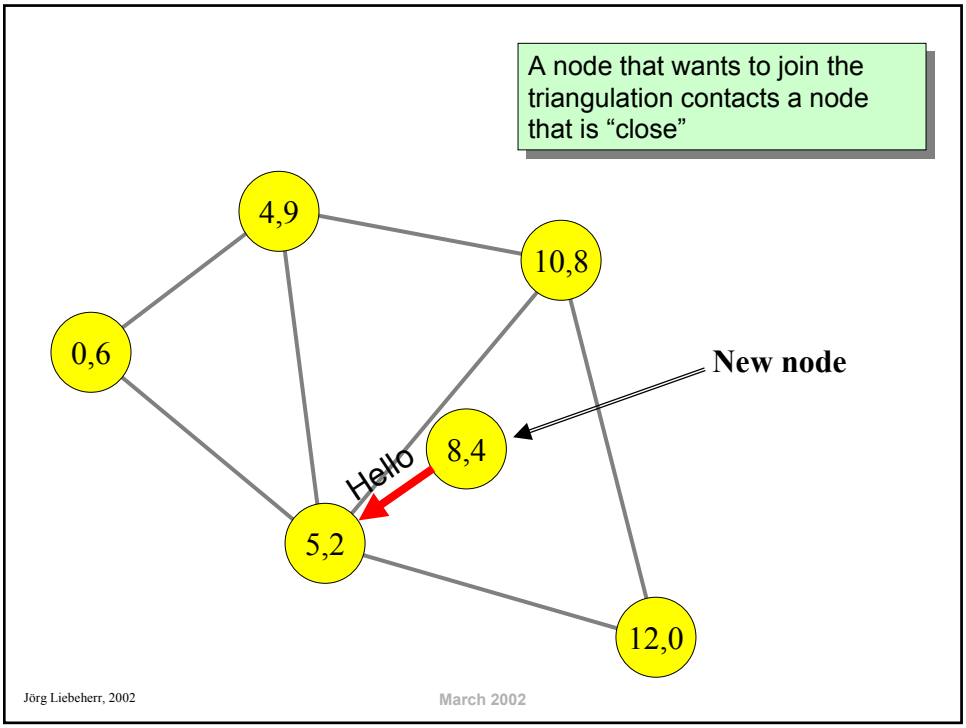


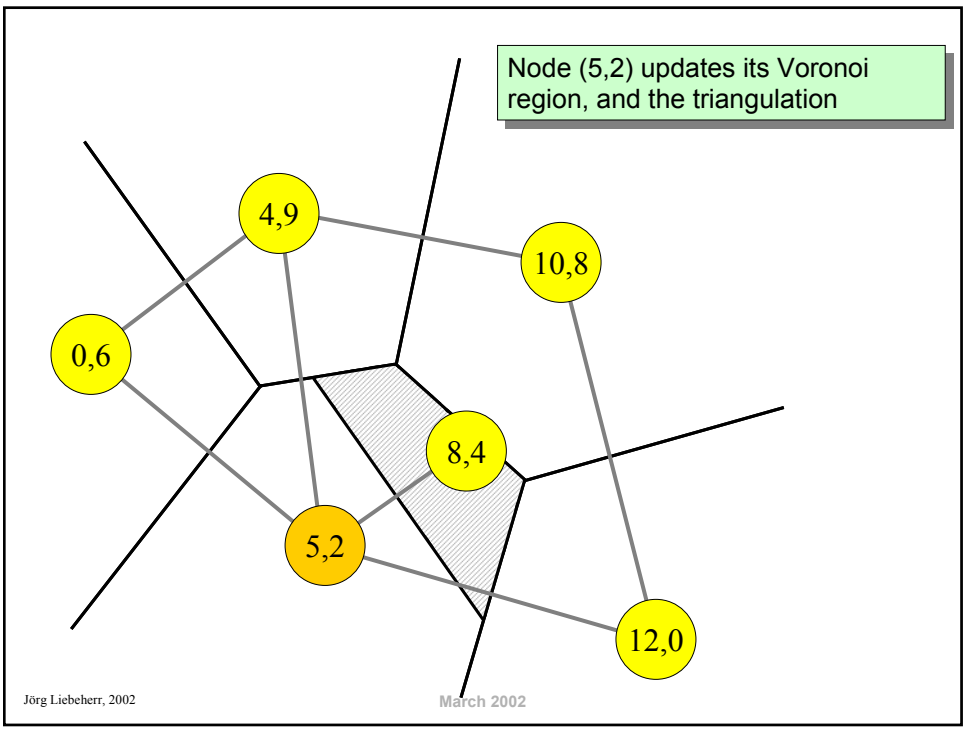

- Each Hello contains the clockwise (CW) and counterclockwise (CCW) neighbors
- Receiver of a Hello runs a "Neighbor test" (→ locally equiangular prop.)
- CW and CCW are used to detect new neighbors

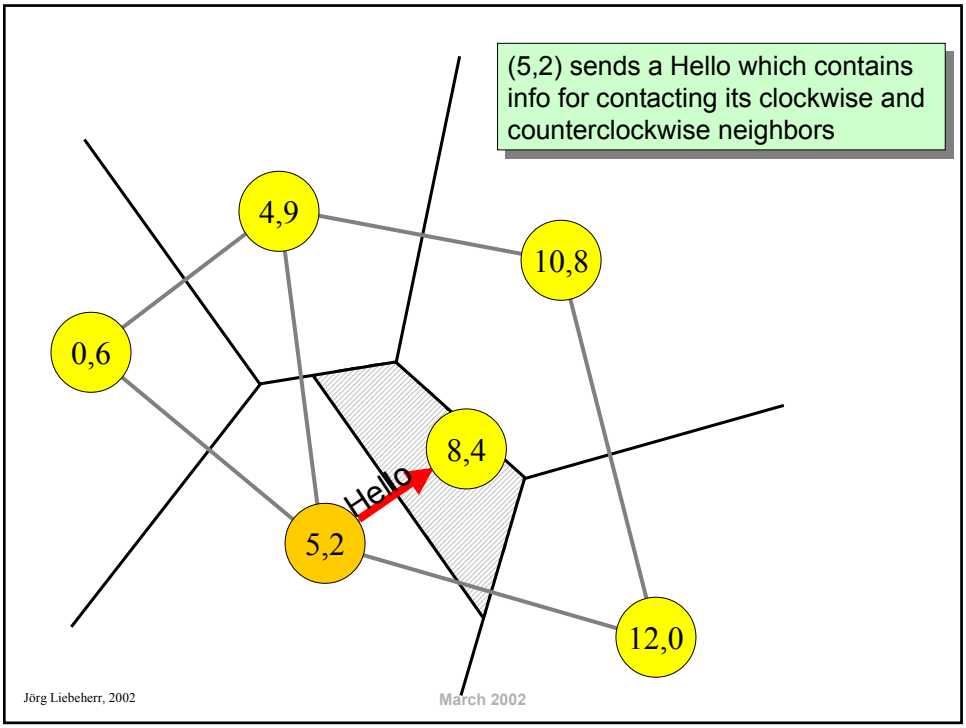


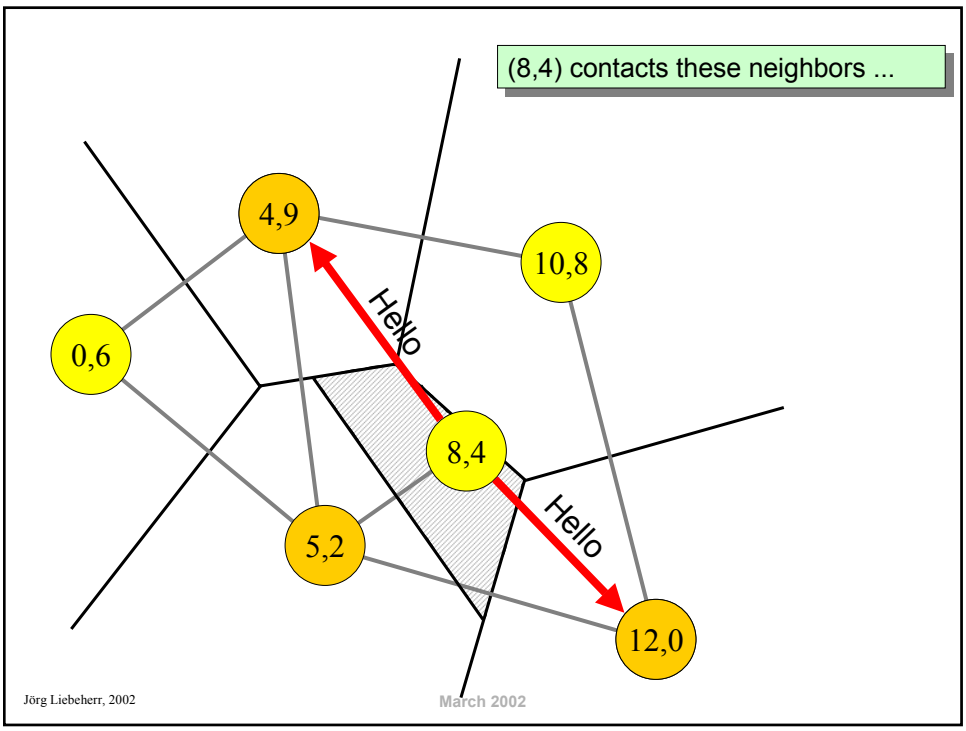

**Neighborhood Table of 10.8**

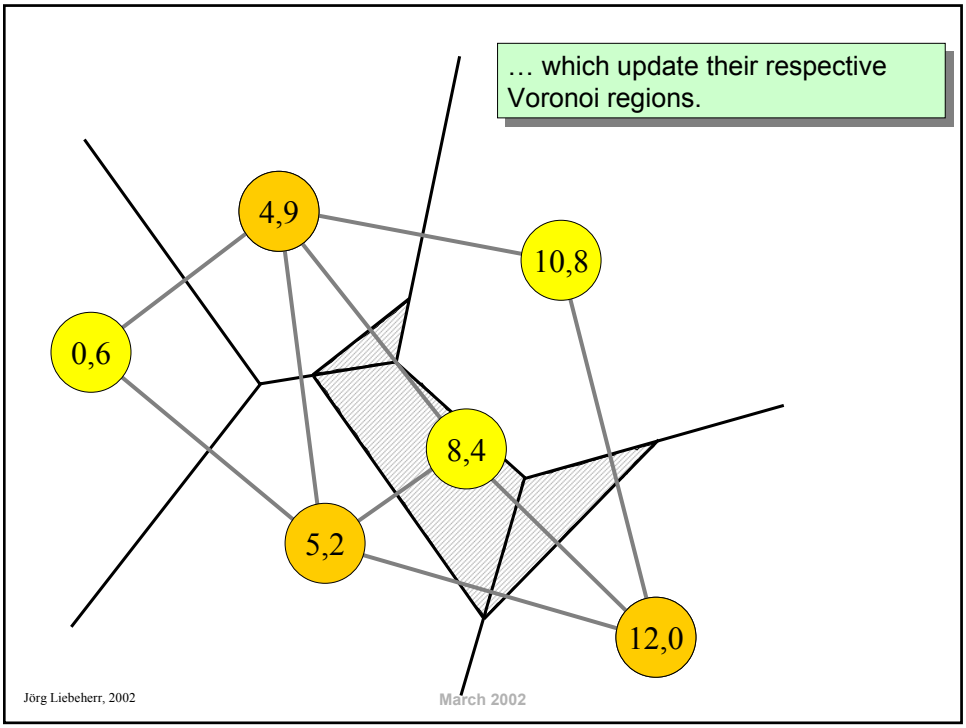
| Neighbor | CW | CCW |
|---|---|---|
| 5,2 | 12,0 | 4,9 |
| 4,9 | 5,2 | – |
| 12,0 | – | 10,8 |

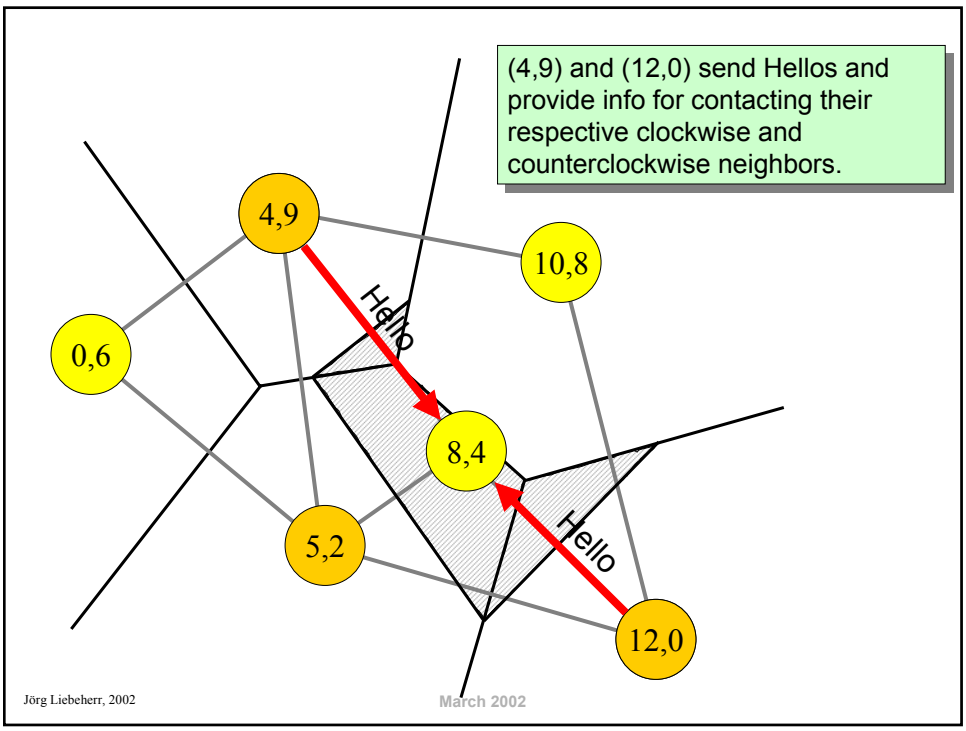

A node that wants to join the triangulation contacts a node that is “close”
4,9
10,8
0,6
New node
8,4
Hello
5,2
12,0
Jörg Liebeherr, 2002
March 2002

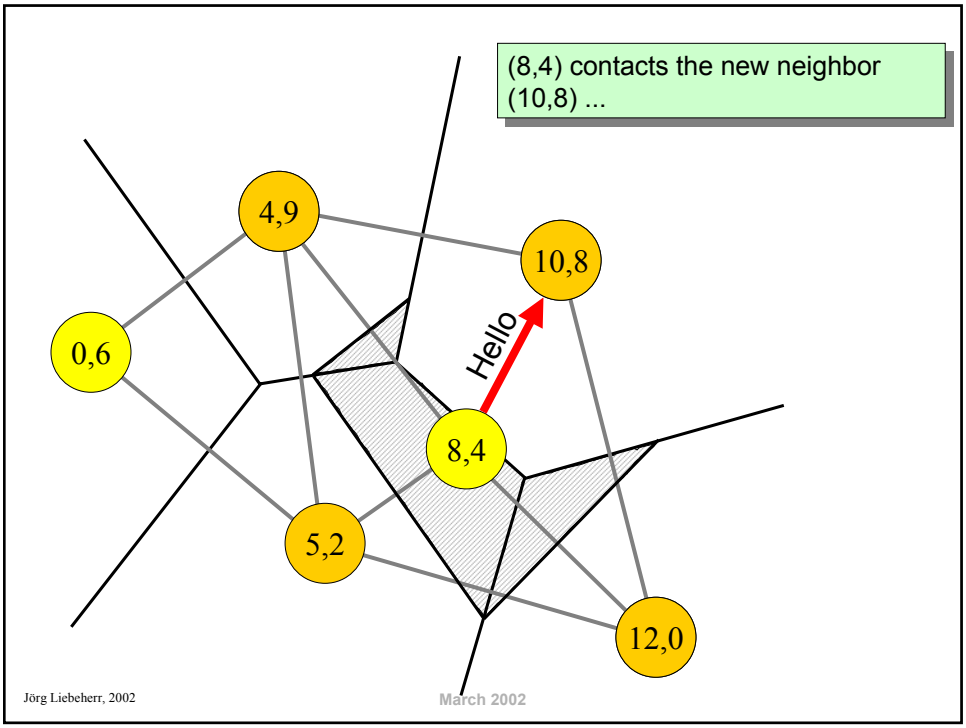

Node (5,2) updates its Voronoi region, and the triangulation
4,9
10,8
0,6
8,4
5,2
12,0
Jörg Liebeherr, 2002
March 2002

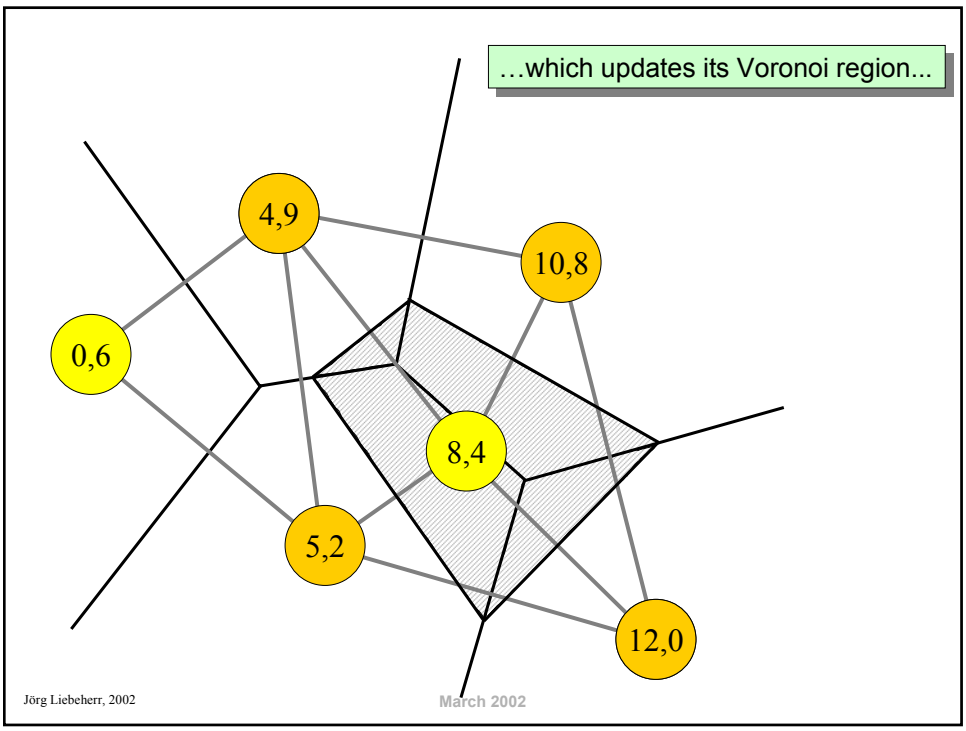

(5,2) sends a Hello which contains info for contacting its clockwise and counterclockwise neighbors
4,9
10,8
0,6
8,4
Hello
5,2
12,0
Jörg Liebeherr, 2002
March 2002

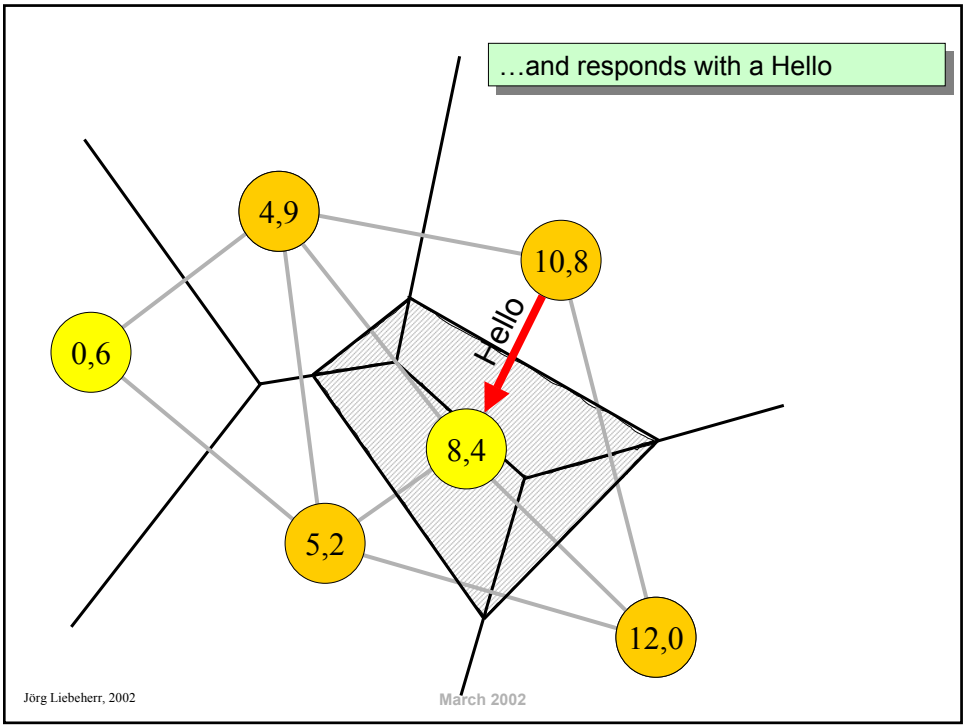

(8,4) contacts these neighbors ...
4,9
10,8
Hello
0,6
8,4
Hello
5,2
12,0
Jörg Liebeherr, 2002
March 2002

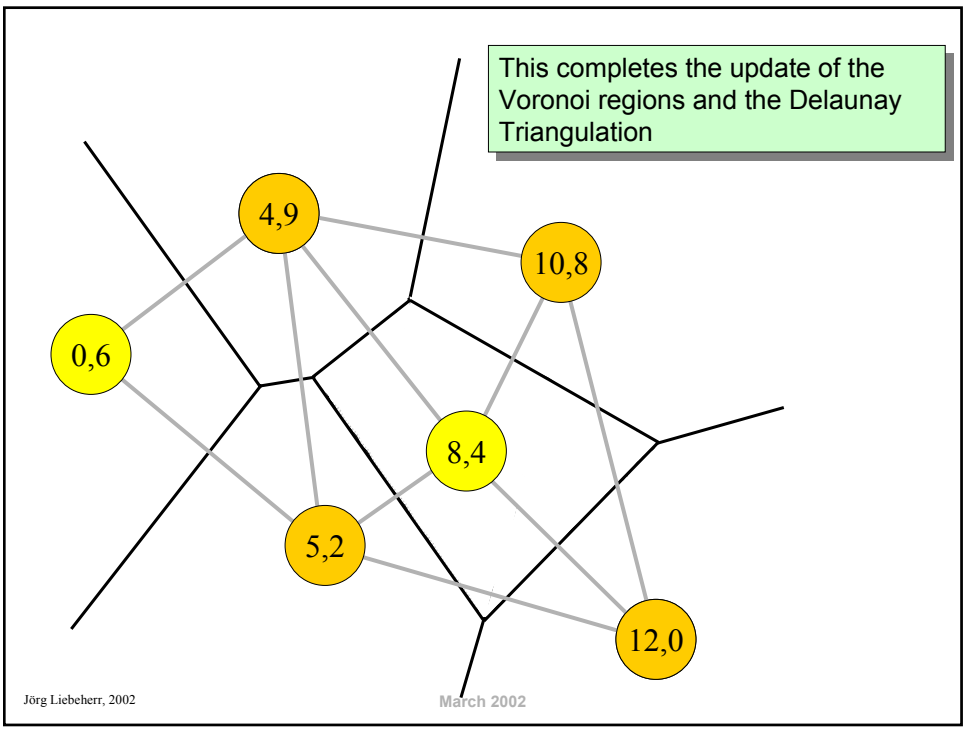

… which update their respective Voronoi regions.
4,9
10,8
0,6
8,4
5,2
12,0
Jörg Liebeherr, 2002
March 2002


(4,9) and (12,0) send Hellos and provide info for contacting their respective clockwise and counterclockwise neighbors.
4,9
10,8
Hello
0,6
8,4
5,2
Hello
12,0
Jörg Liebeherr, 2002
March 2002


(8,4) contacts the new neighbor (10,8) ...
4,9
10,8
0,6
Hello
8,4
5,2
12,0
Jörg Liebeherr, 2002
March 2002


…which updates its Voronoi region...
4,9
10,8
0,6
8,4
5,2
12,0
Jörg Liebeherr, 2002
March 2002


…and responds with a Hello
4,9
10,8
Hello
0,6
8,4
5,2
12,0
Jörg Liebeherr, 2002
March 2002


This completes the update of the Voronoi regions and the Delaunay Triangulation
4,9
10,8
0,6
8,4
5,2
12,0
Jörg Liebeherr, 2002
March 2002

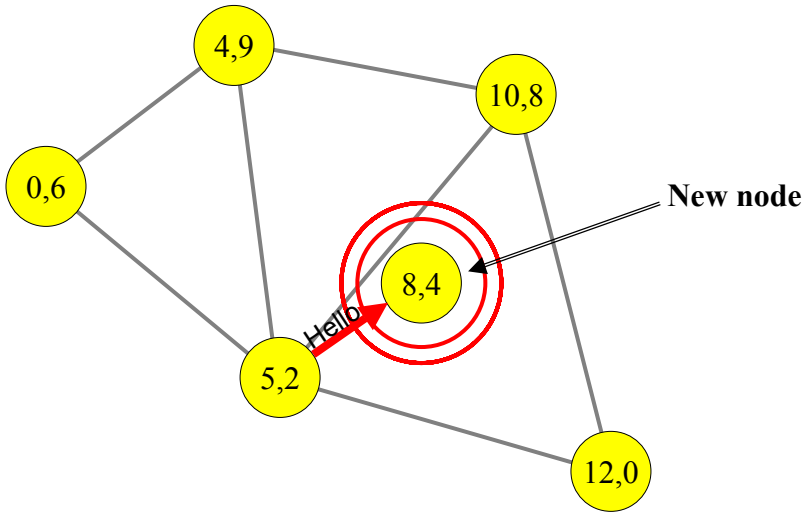
# Rendezvous Methods

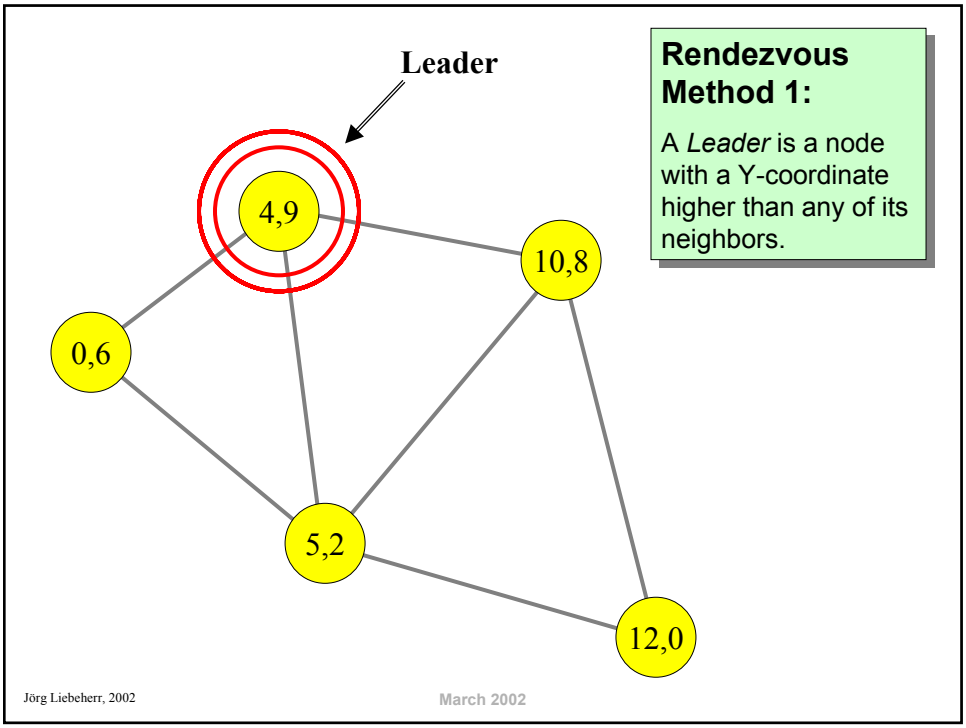
- **Rendezvous Problems**:
  - How does a new node detect a member of the overlay?
  - How does the overlay repair a partition?

- **Three solutions**:
  1. Announcement via broadcast
  2. Use of a rendezvous server
  3. Use `likely' members ("Buddy List")

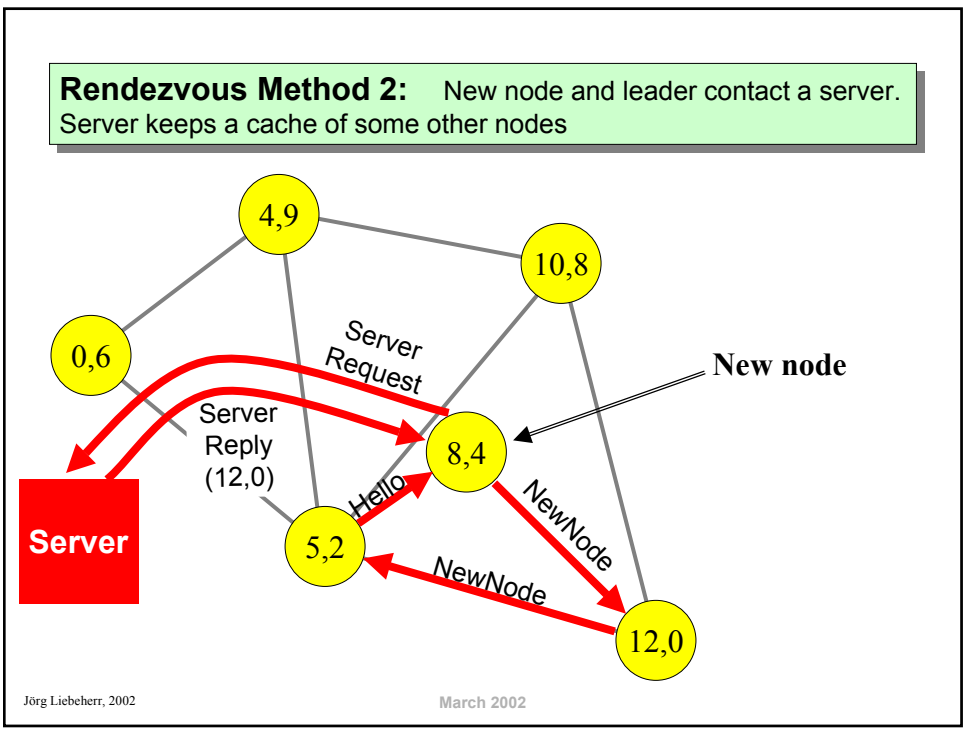


**Rendezvous Method 1:** Announcement via broadcast (e.g., using IP Multicast)

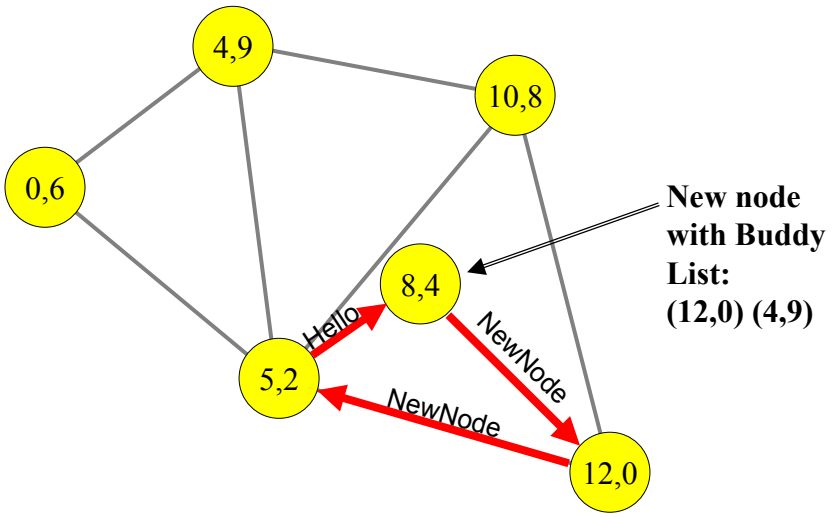



Leader
4,9
10,8
0,6
5,2
12,0
Rendezvous Method 1:
A *Leader* is a node with a Y-coordinate higher than any of its neighbors.



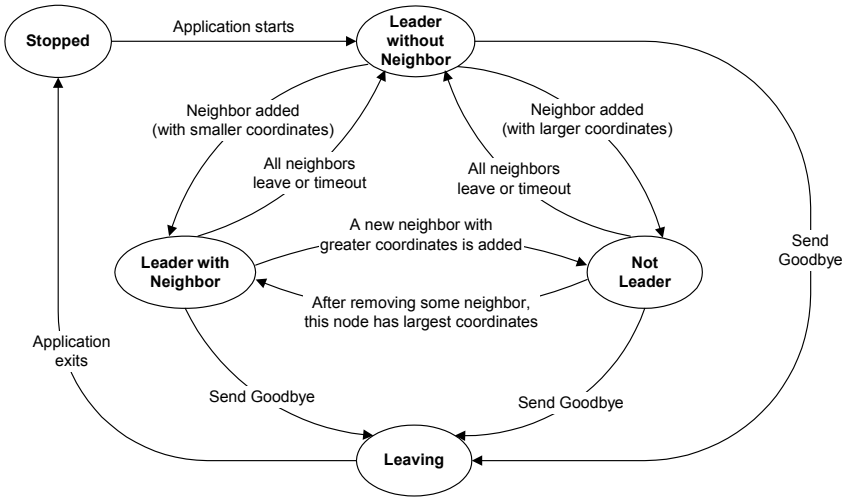

Rendezvous Method 2: New node and leader contact a server. Server keeps a cache of some other nodes
4,9
10,8
0,6
Server Request
New node
Server Reply (12,0)
8,4
Server
Hello
NewNode
5,2
NewNode
12,0

**Rendezvous Method 3:** Each node has a list of "likely" members of the overlay network

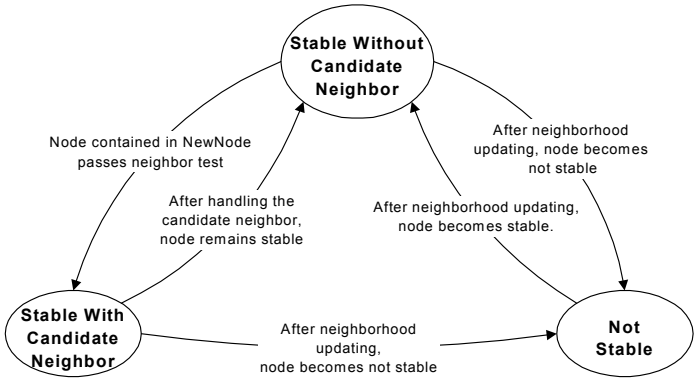





# State Diagram of a Node



# Sub-states of a Node

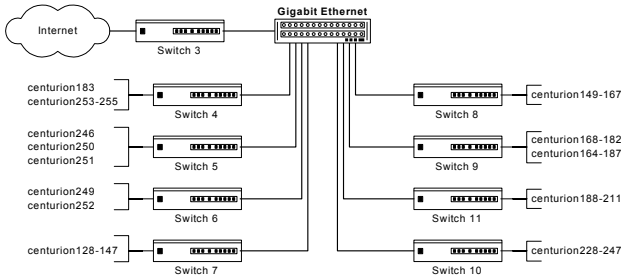



• A node is **stable** when all nodes that appear in the CW and CCW neighbor columns of the neighborhood table also appear in the neighbor column



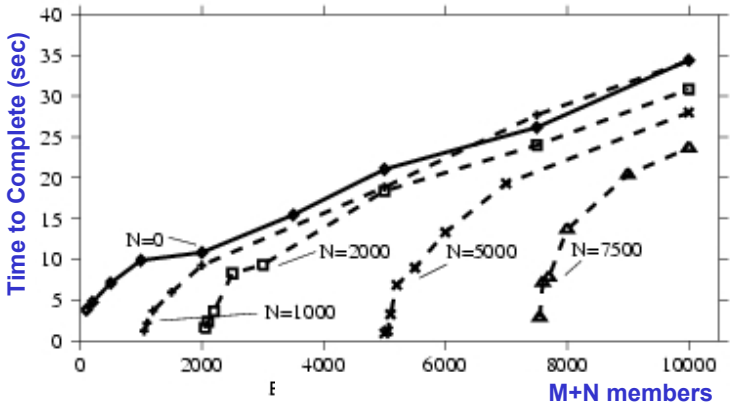
# Measurement Experiments

- **Experimental Platform:**
  Centurion cluster at UVA (cluster of 300 Linux PCs)
  - **2 to 10,000 overlay members**
  - **1–100 members per PC**



## Experiment: Adding Members

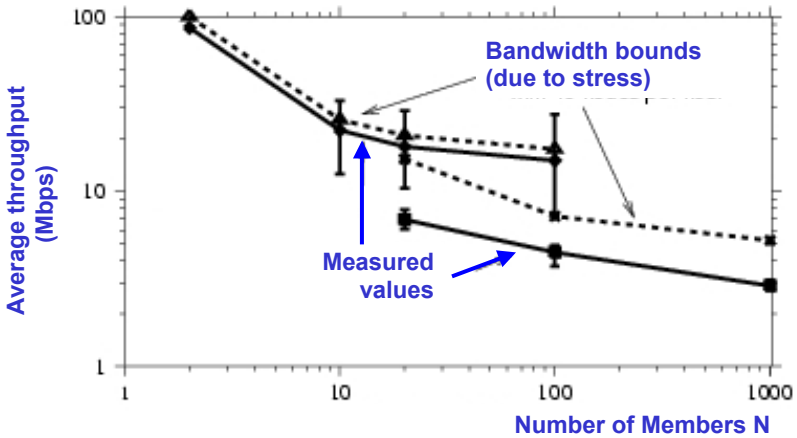
How long does it take to add M members to an overlay network of N members ?




Jörg Liebeherr, 2002

## Experiment: Throughput of Multicasting

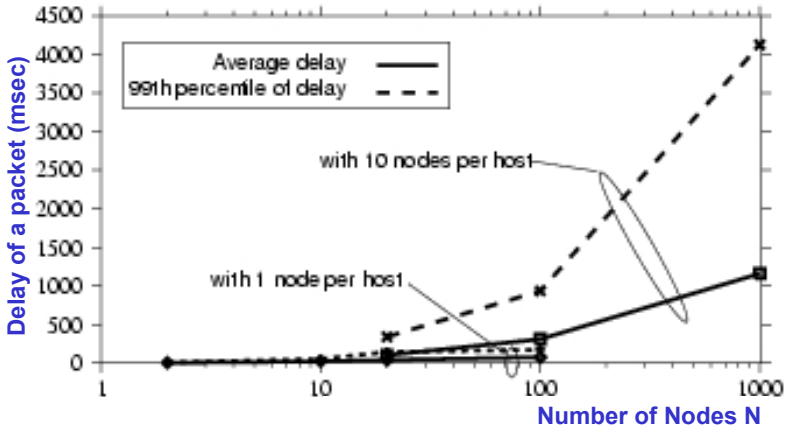
100 MB bulk transfer for N=2-100 members (1 node per PC)
10 MB bulk transfer for N=20-1000 members (10 nodes per PC)




Jörg Liebeherr,

## Experiment: Delay

100 MB bulk transfer for N=2-100 members (1 node per PC)
10 MB bulk transfer for N=20-1000 members (10 nodes per PC)






## Summary

- Use of Delaunay triangulations for overlay networks
- Delaunay triangulation observes ‘coordinates” but ignores network topology
- No routing protocol is needed in the overlay

- **Ongoing efforts:**
  - Use delay measurements to determine coordinates
  - HyperCast on handheld devices (iPaQs)
  - Enhance data services: “Message Store”

- HyperCast Project website:
  **http://www.cs.virginia.edu/~hypercast**