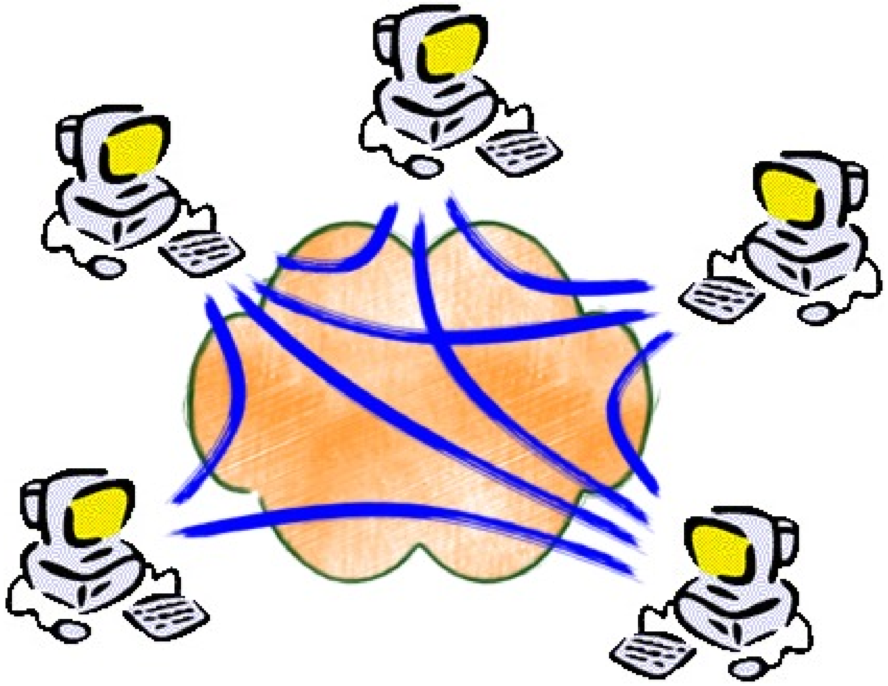
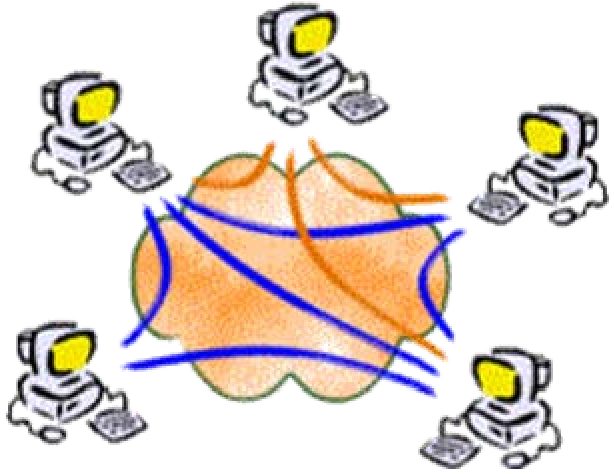
HyperCast

Jorg Liebeherr
University of Virginia



Acknowledgements

- Developed in my research group since 1999
- Contributors:
  - *Past: Bhupinder Sethi, Tyler Beam, Burton Filstrup, Mike Nahas, Dongwen Wang, Konrad Lorincz, Jean Ablutz, Haiyong Wang, Weisheng Si, Huafeng Lu, Josh Zaritsky, Guimin Zhang, Jianping Wang, Guangyu Dong, Greg Mattes, Wittawat Tantisiriroj*

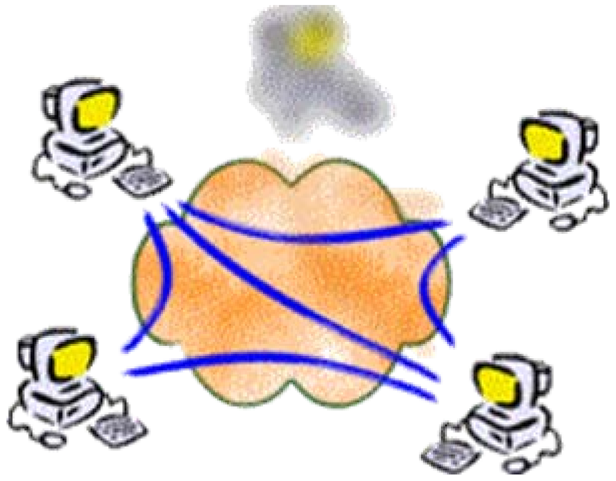


- Supported in part by the National Science Foundation:

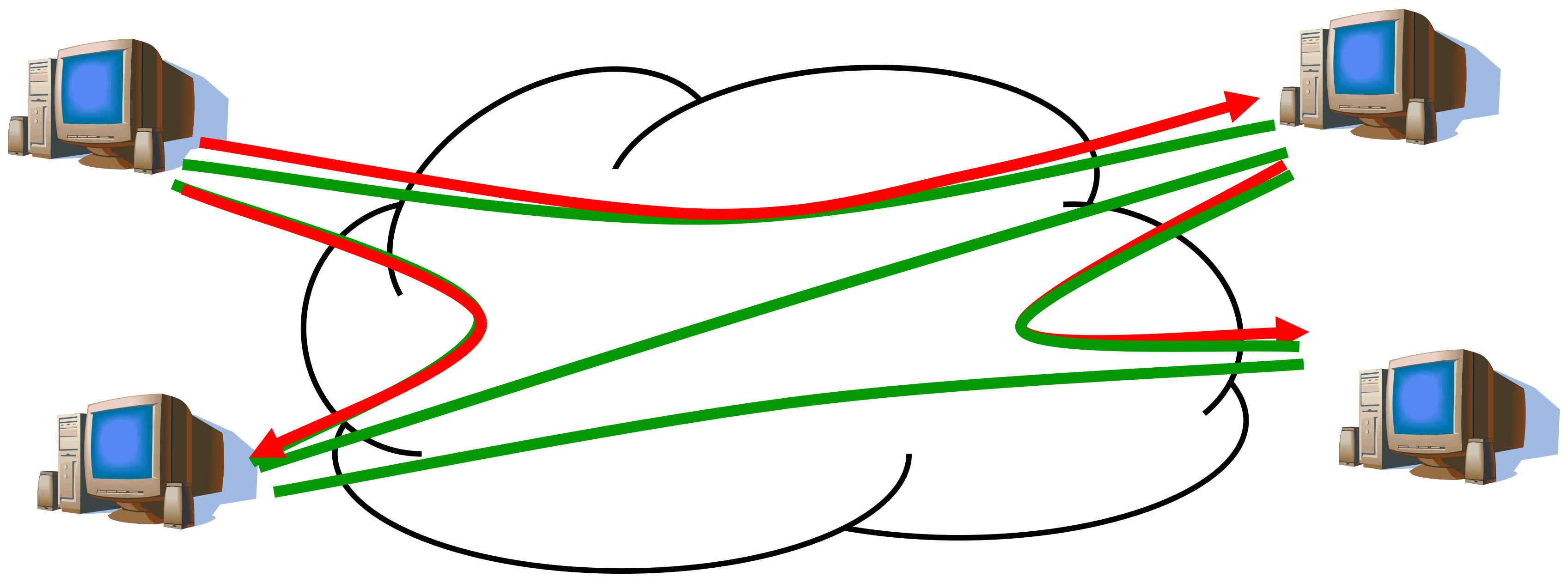
D E N A L I

HyperCast

- Enables information exchange groups (application layer overlays) at the application layer over substrate networks software with:
  - Dynamically changing group membership
  - Arbitrarily many groups of arbitrary size
  - Support of security needs
  - Monitor and Control functions



Overlays



- Overlay is viewed as a network of application programs
- Data is exchanged data over a substrate network (Internet, ad-hoc, sensor network)

Topics

- Joining and creating an overlay
- Data exchange
- Security features
- Monitoring and control of overlays

A simple HyperCast program

```
//Generate the configuration object
OverlaySocketConfig ConfObj =
          OverlaySocketConfig.createOLConfig("hypercast.xml");

//Create a socket
 I_OverlaySocket socket=ConfObj.createOverlaySocket(null);

//Join the group
socket.joinOverlay();

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

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

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

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

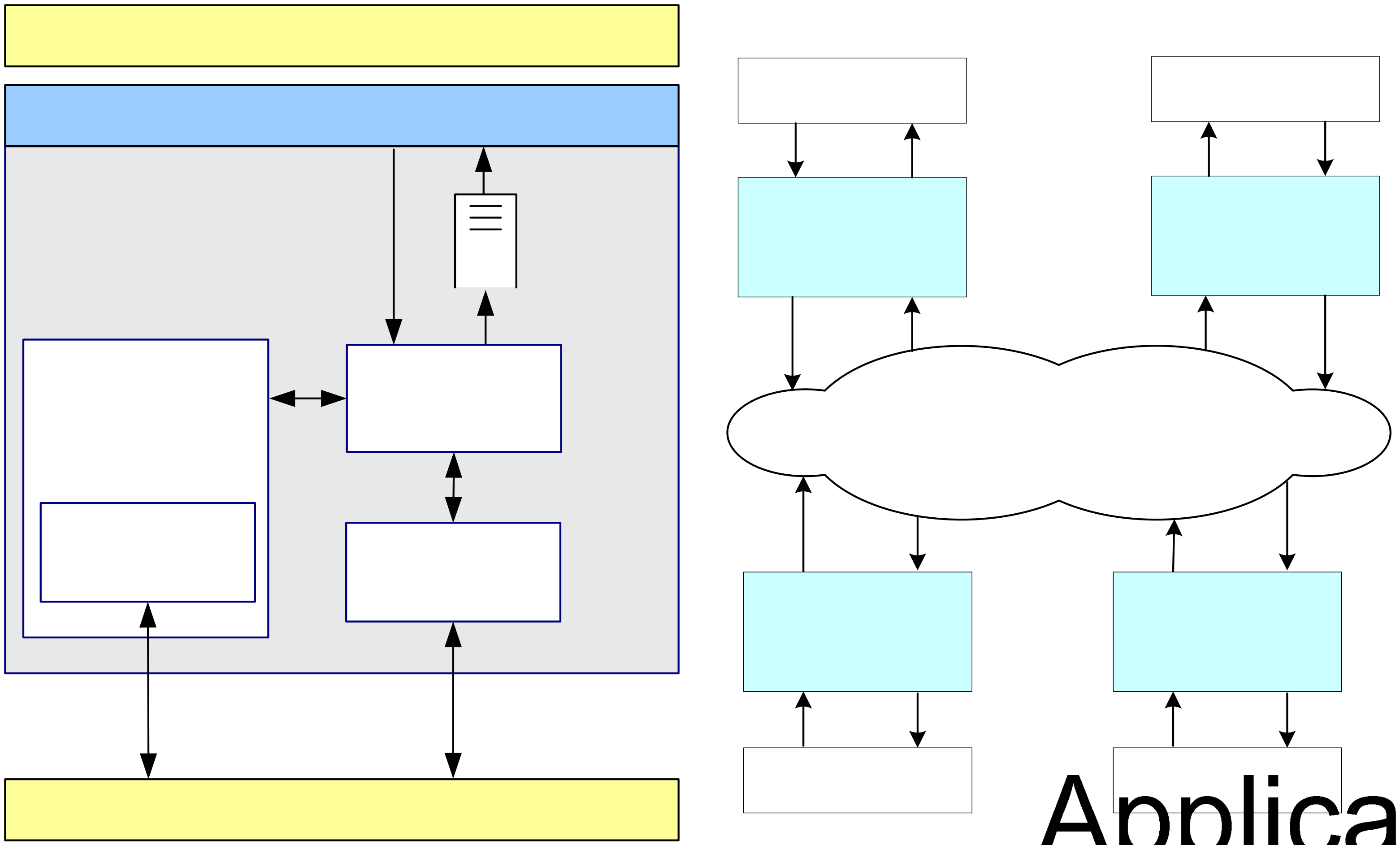
Configuration file specifies parameters for an overlay

Creates an inter-face to the overlay ("socket")

Joins the overlay

Sends to overlay

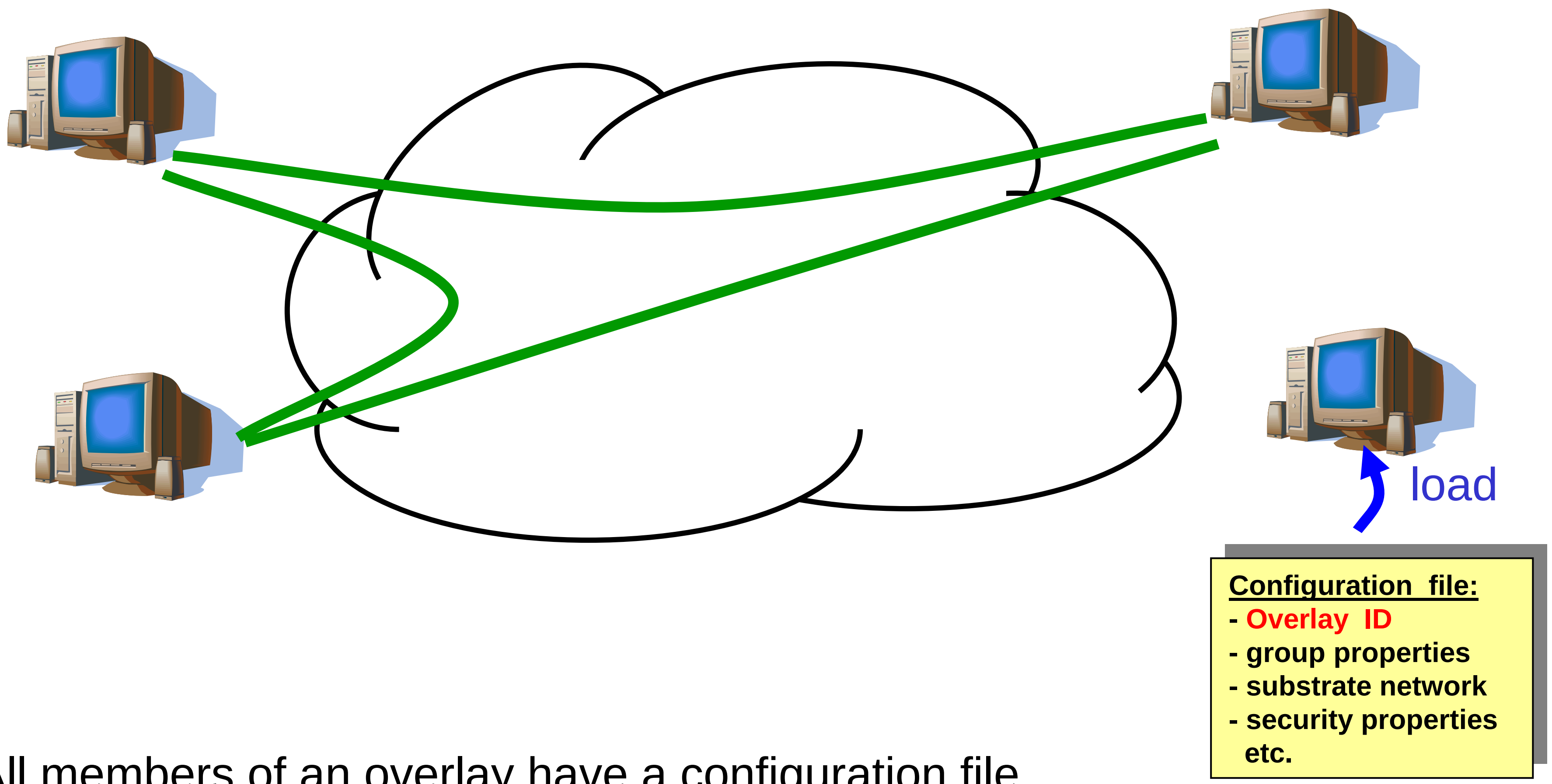
Overlay Socket and Overlay Network



Applicatior

Joining and Creating overlay networks

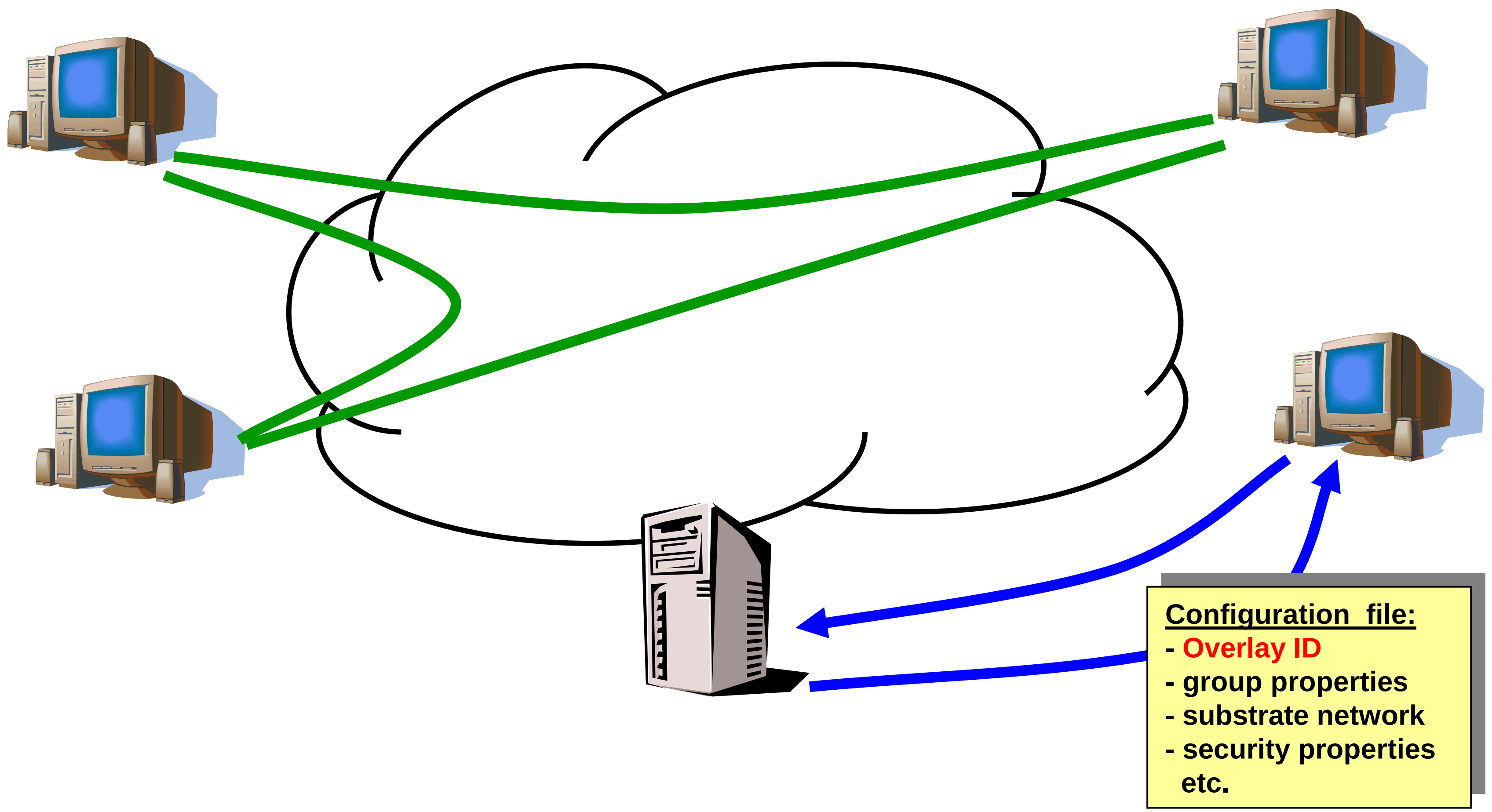
Joining an overlay: Configuration File



<u>Configuration file:</u>
- **Overlay ID**
- **group properties**
- **substrate network**
- **security properties etc.**

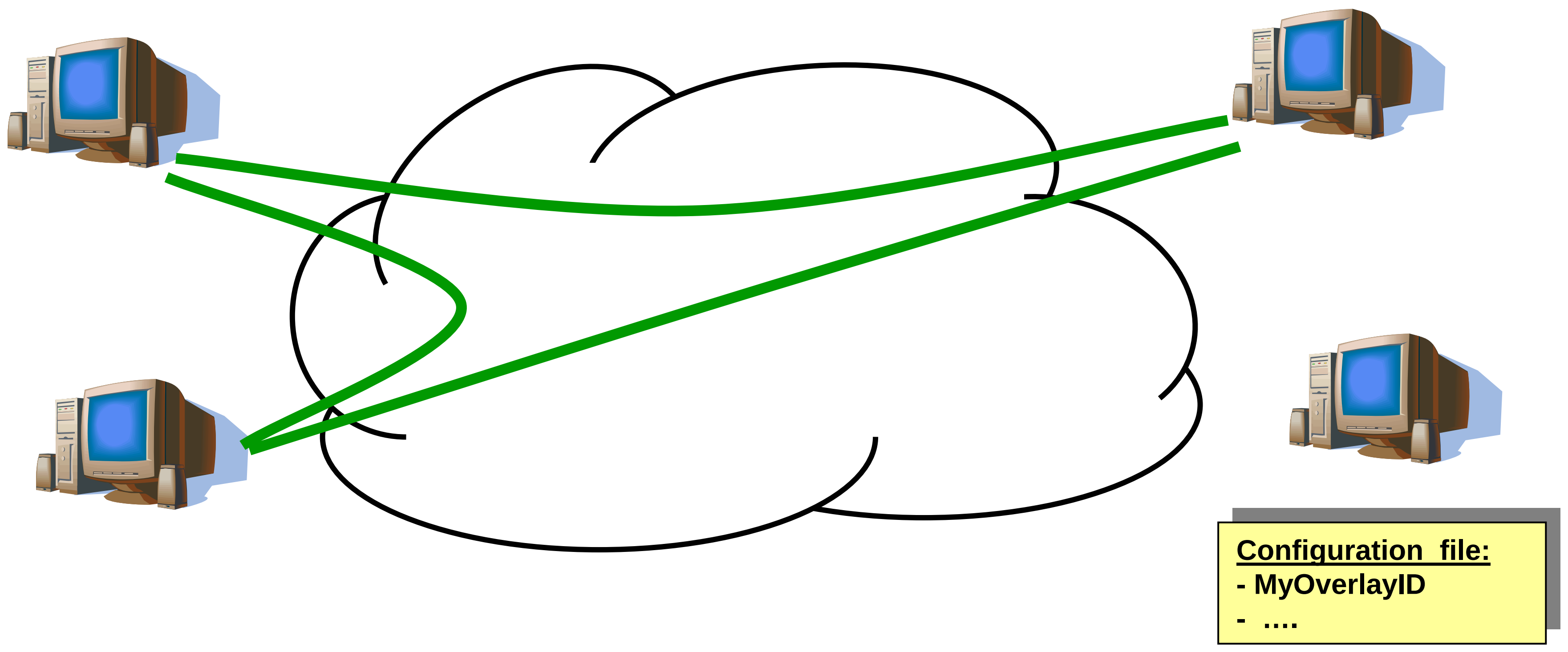
- All members of an overlay have a configuration file
- Configuration file specifies which overlay to join and how to join it
- Configuration file is distributed in advance ….

Joining an overlay: Configuration File



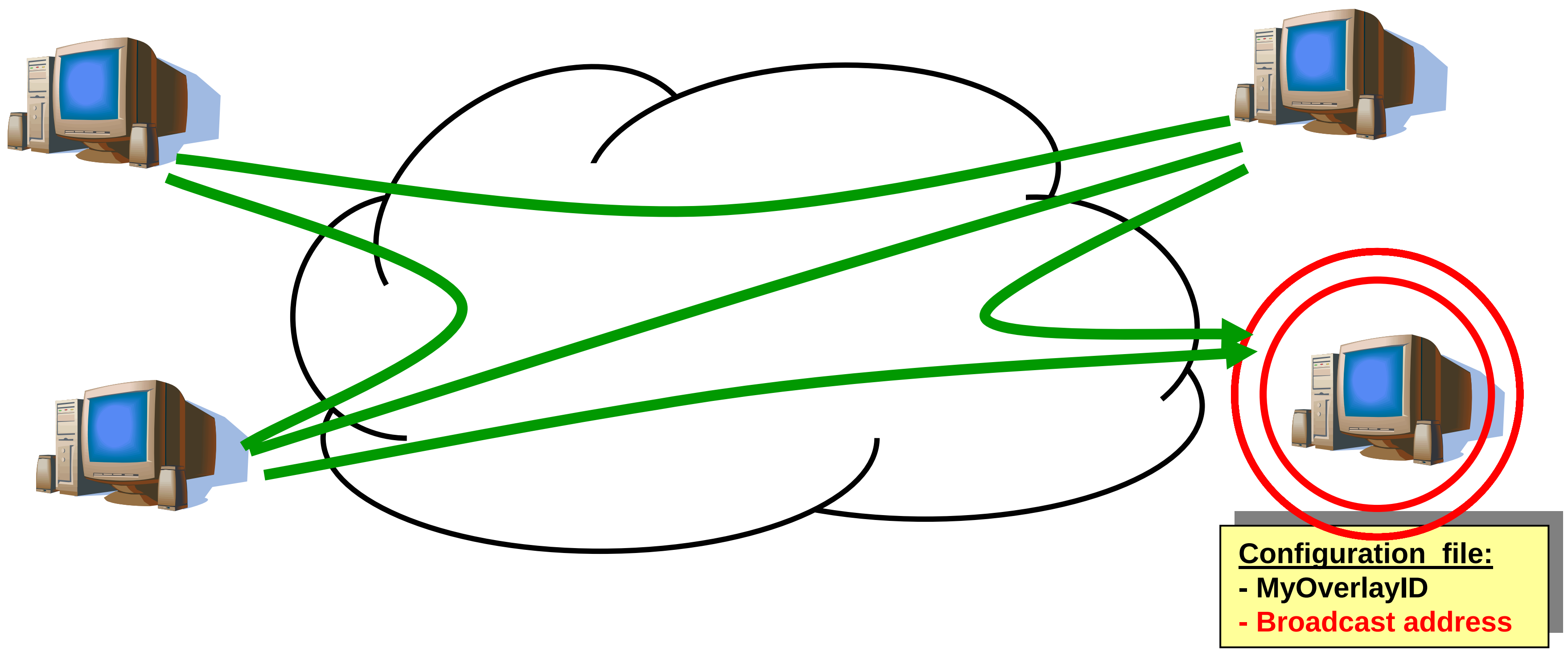
- … or configuration is downloaded from a server

Joining an overlay: Contacting group members



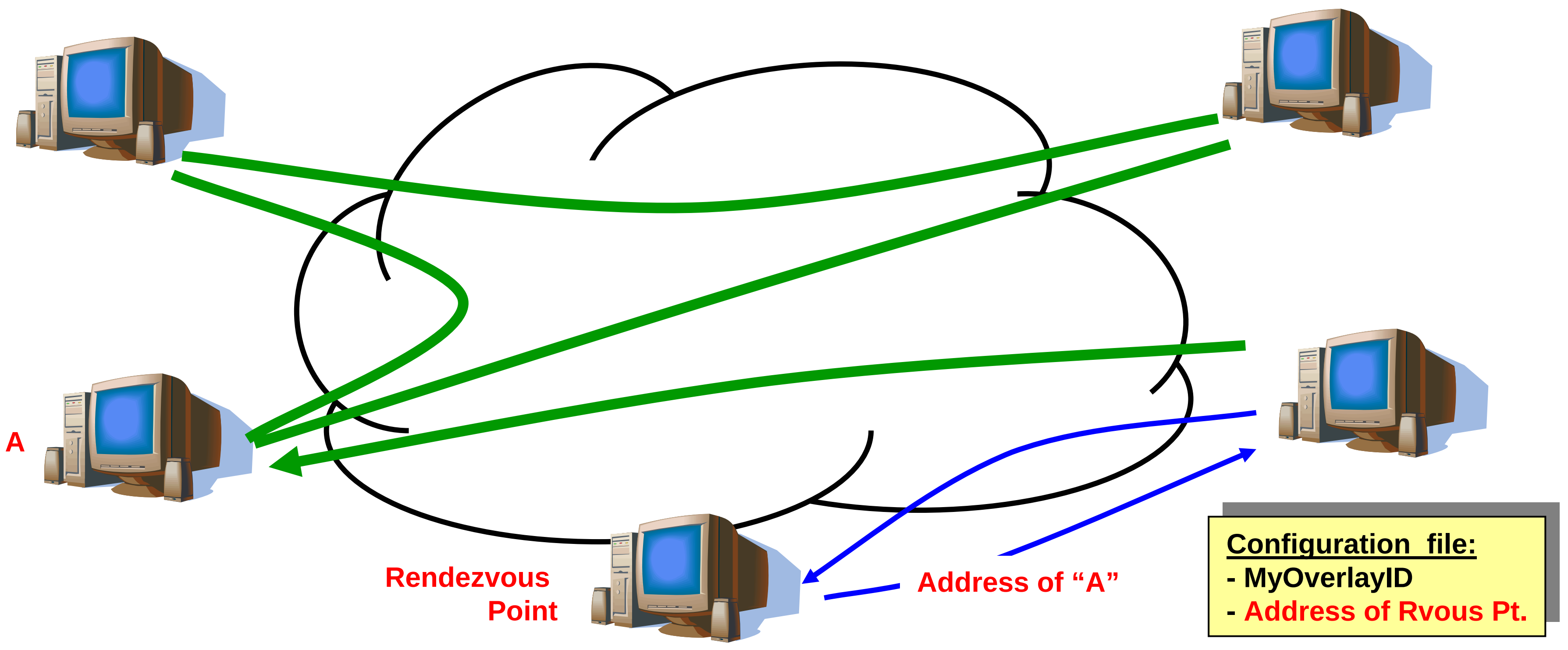
- There are three methods by which a new member can contact overlay members:
  1. **Announcement via broadcast**
  2. **Dedicated Rendezvous Point**
  3. **Contact well-known members (“buddies”)**

Joining an overlay: Announcement via broadcast



- If an application can send broadcast messages, it can announce its presence
- Existing members that receive the broadcast contact the new member
- Broadcast address is stored in configuration file

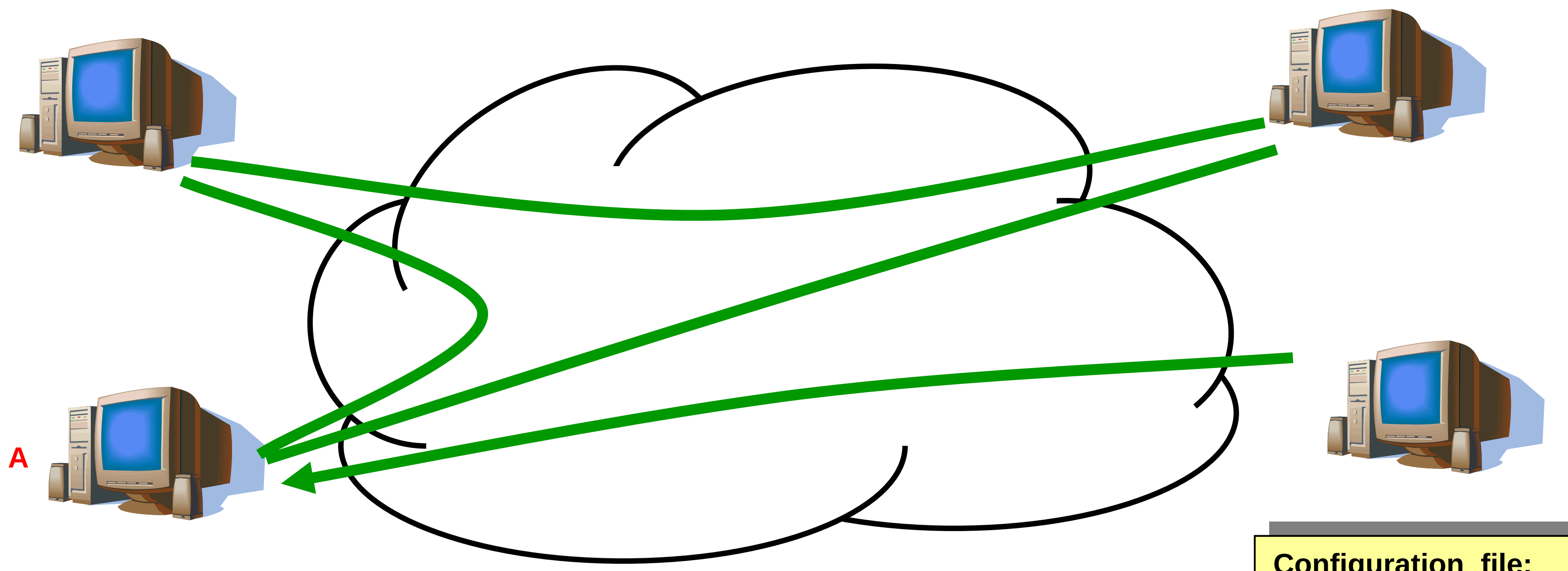
Joining an overlay: Rendezvous Point



Configuration file:
- MyOverlayID
- Address of Rvous Pt.

- There is a dedicated application that acts as rendezvous point
- Rendezvous point maintains a list of some current members
- New application gets a current member from the rendezvous point
- Address of rendezvous point is in configuration file

Joining an overlay: Contact well-known members



Configuration file:
- **MyOverlayID**
- **Address of buddy "A".**

- Address of some current members is in configuration file ("buddy list")
- New application contacts the members in the list

Creating a new overlay

- Create a new or modify an existing configuration file
  - Select an overlay identifier
  - Change properties of the overlay socket

Configuration file:
- MyOverlayID
- group properties
- substrate network
- security properties

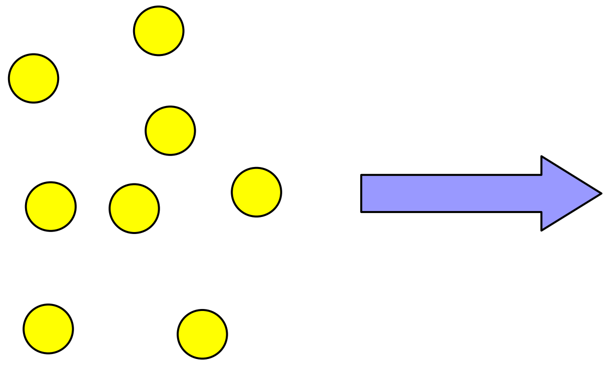
- Have the application program read the new file

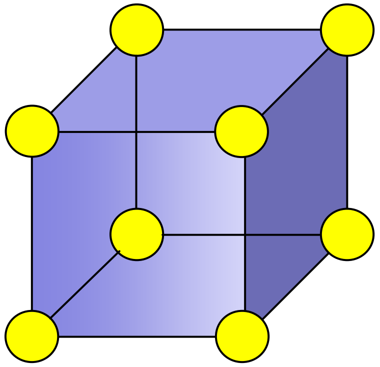
```
OverlaySocketConfig ConfObj =
OverlaySocketConfig.createConfig(“MyConfiguration.xml");
```

- Distribute the configuration file:
  - Out-of-band, or
  - Overlay server

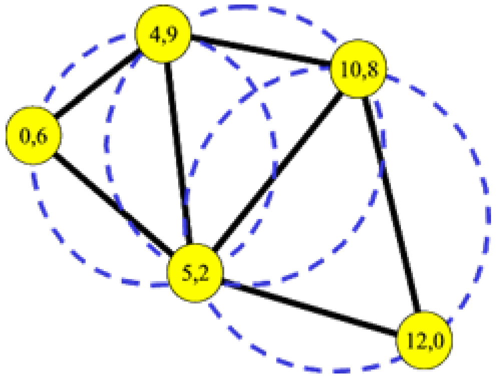
Overlay Network Topologies

- **Overlay can be organized in a variety of topologies**

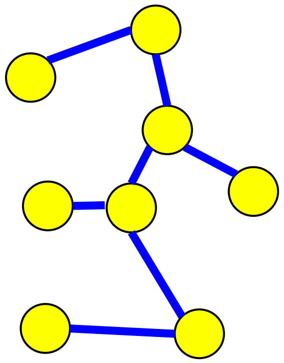




Hypercube, DHT



Triangulation

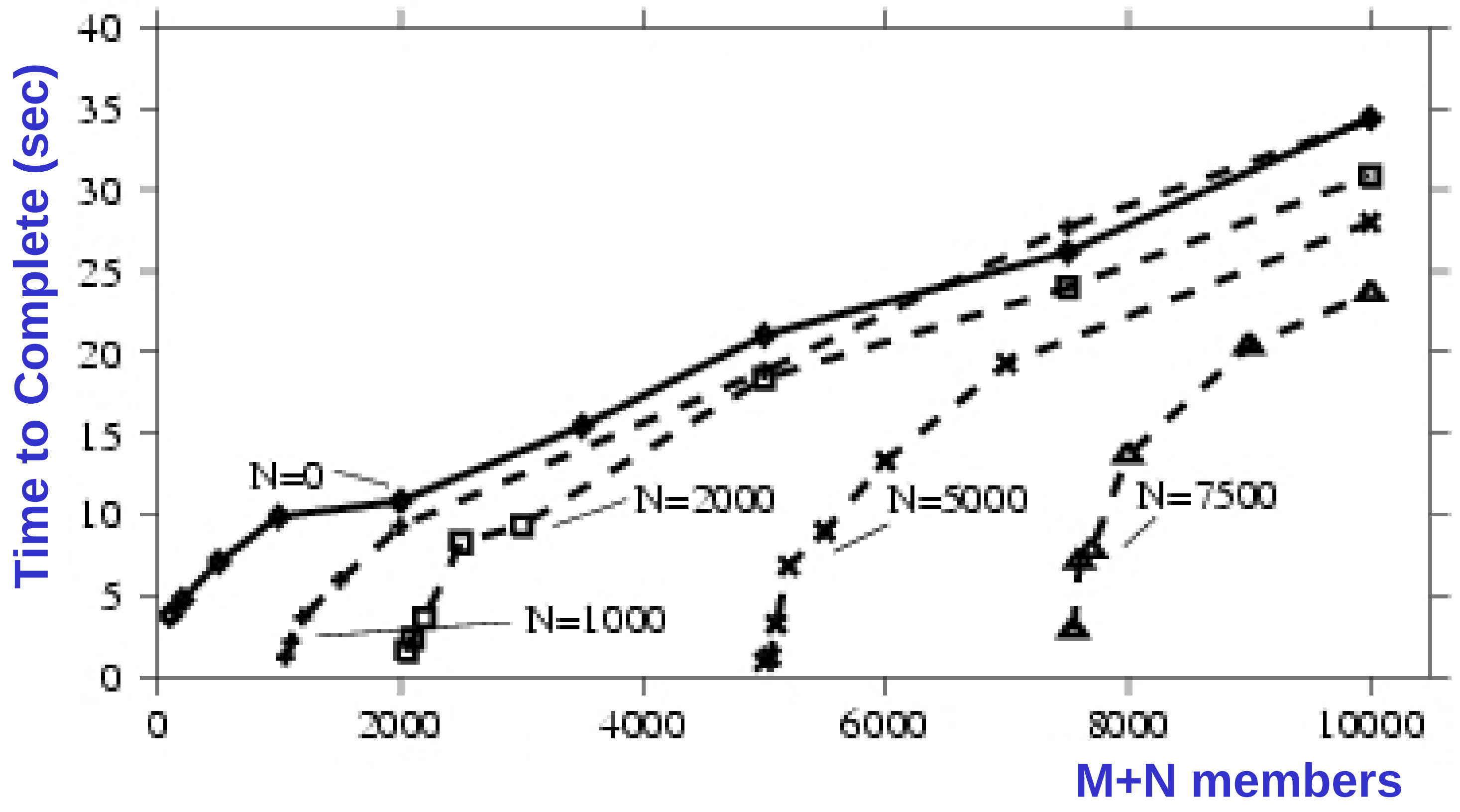


Spanning tree

(for mobile ad hoc)

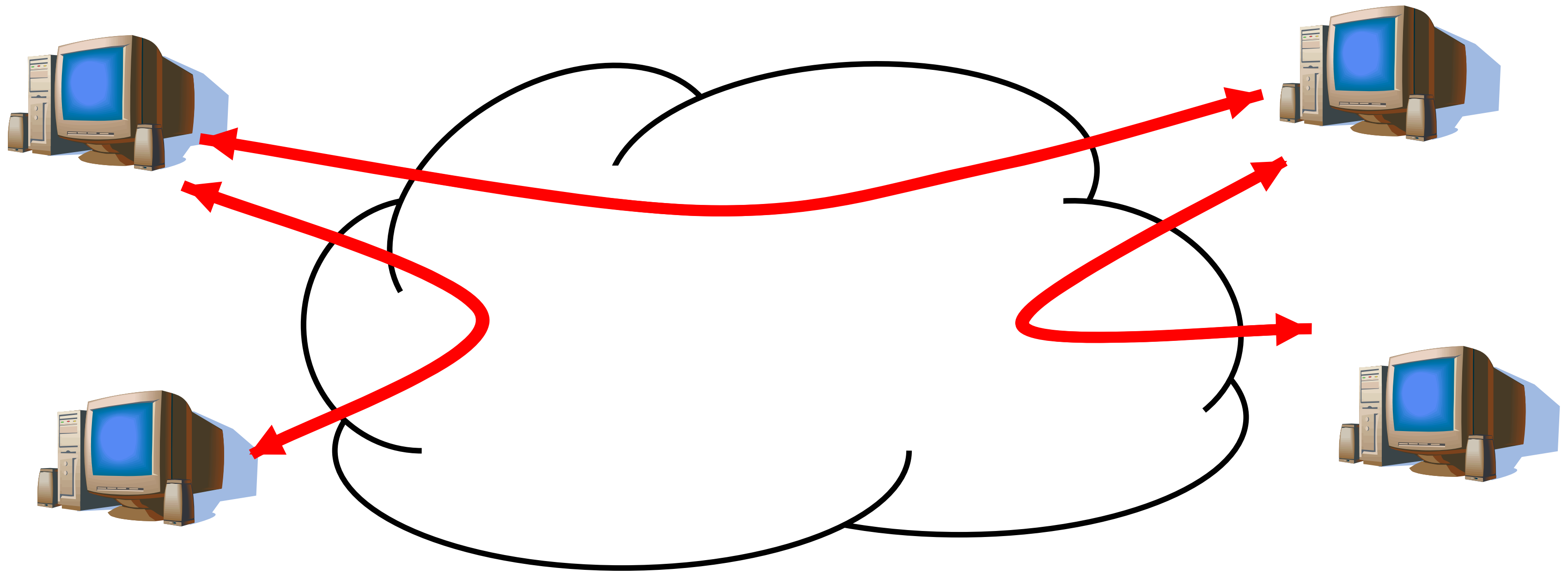
How fast can we built an overlay?

- Measurements of a cluster of 100 Linux PCs
- **Experiment:** Add M members to an overlay network of N members:



Data exchange

Data exchange

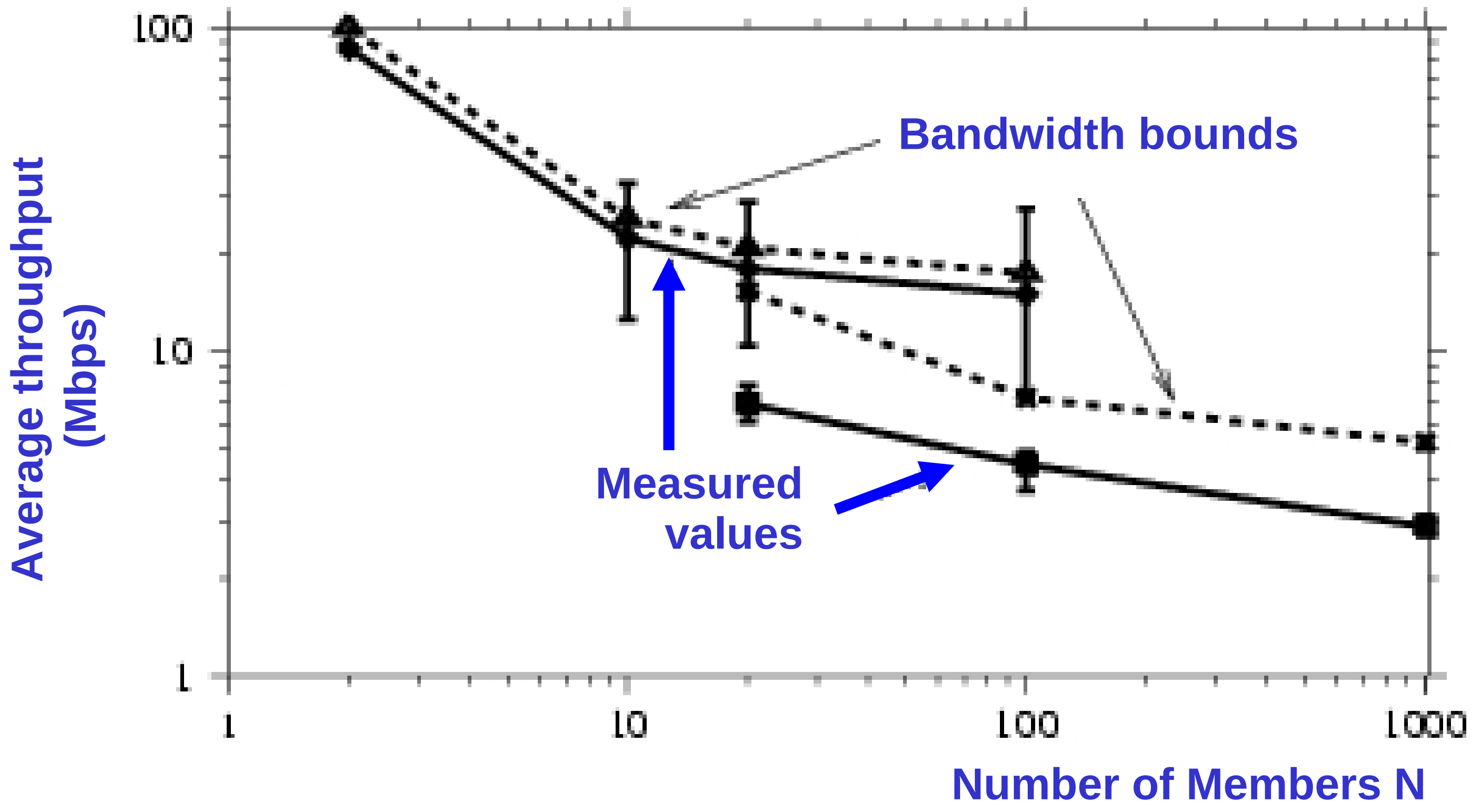


- Several data exchanges are supported:
  1. **One-to-One (Unicast)**
  2. **One-to-All (Multicast)**
  3. **All-to-One (Incast)**

- Delivery Semantics:
  1. **Best-effort**
  2. **In-order**
  3. **Reliable**

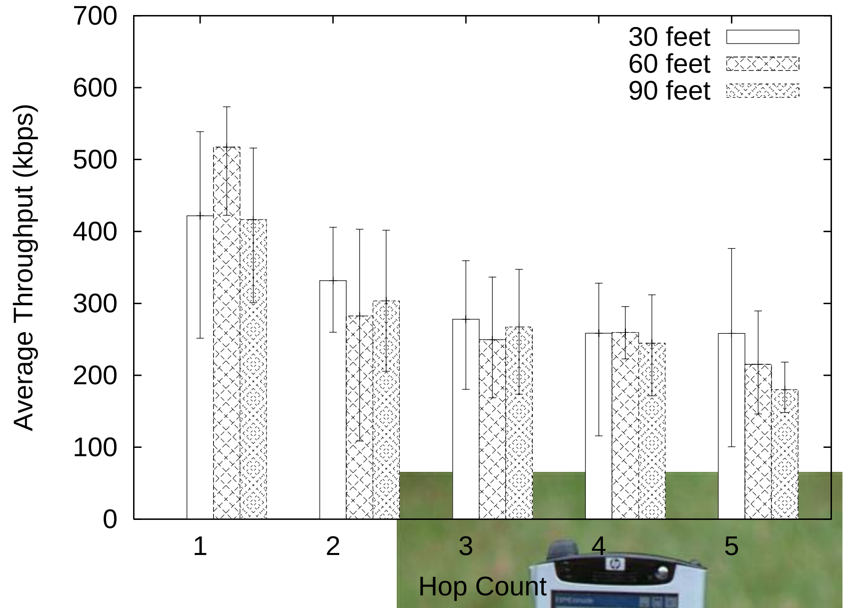
How much data can we send?

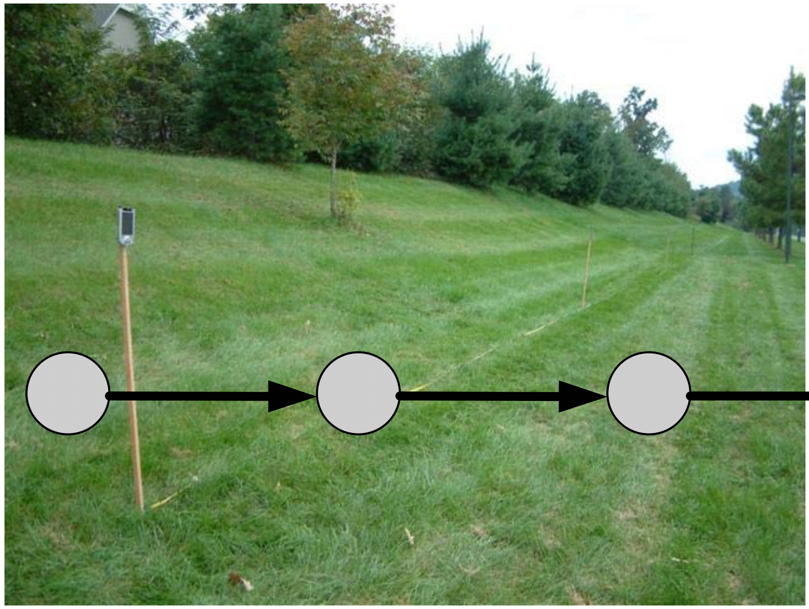
Bulk data transfer from 1 sender to 2-1000 receivers:
100 MB bulk transfer for N=2-100 members (1 node per PC)
10 MB bulk transfer for N=20-1000 members (10 nodes per PC)

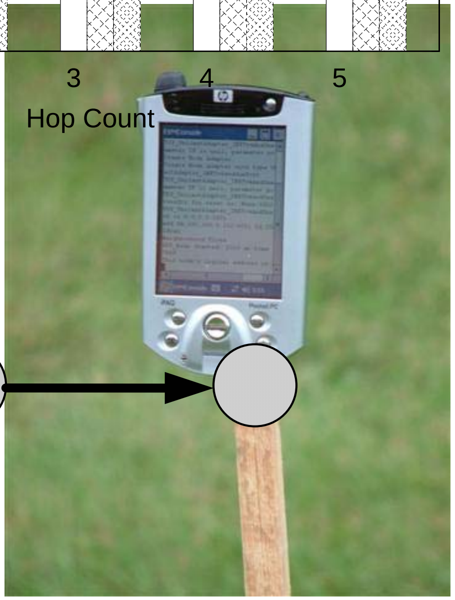


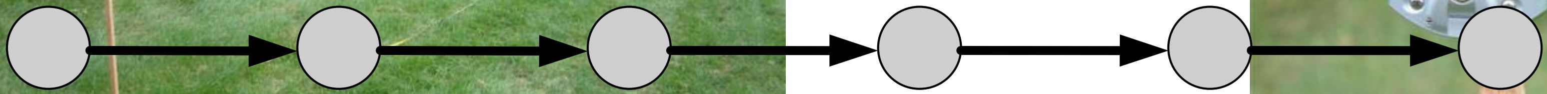
Performance: HyperCast on PDAs

- **Multihop performance**
  - Setting: six iPAQ PDA in a line
  - Substrate network: TCP adapter
  - Send unicast message in greedy fashion
  - Topology is fixed
  - Distance between node is varied









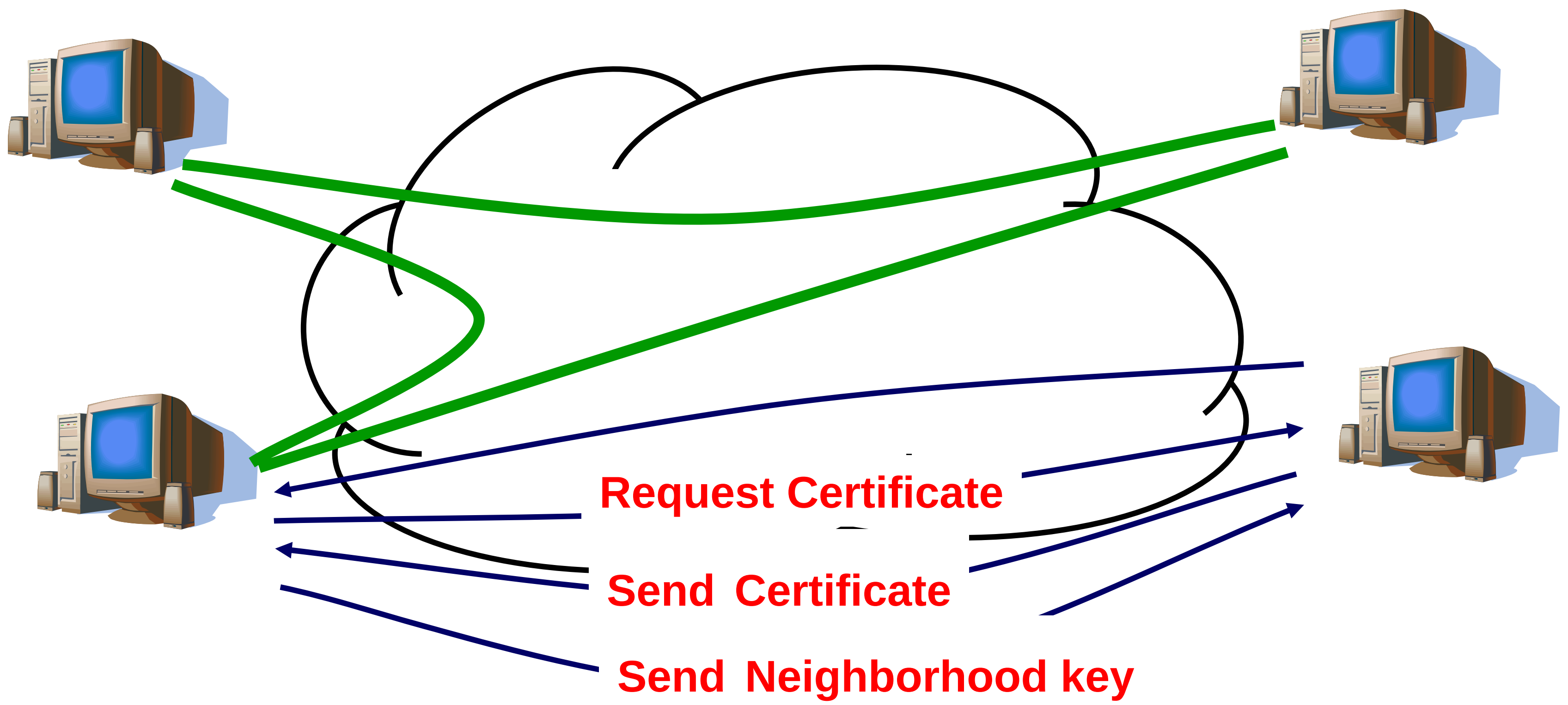
Security features

Security Goals

- *Backward secrecy*
  - A new member should not be able to access data transmitted before the member joined
- *Forward secrecy*
  - A member cannot access data that is transmitted after the member left

- Approach: **Neighborhood Key**
  - Each member maintains a secret key (“neighborhood key”) that it shares with its neighbors

Key Management



- A new members must present a signed certificate to each member that is contacted for the first time
- Once authenticated, it obtains a **neighborhood key** from each neighbor
- A member generates a new key each time its neighborhood changes

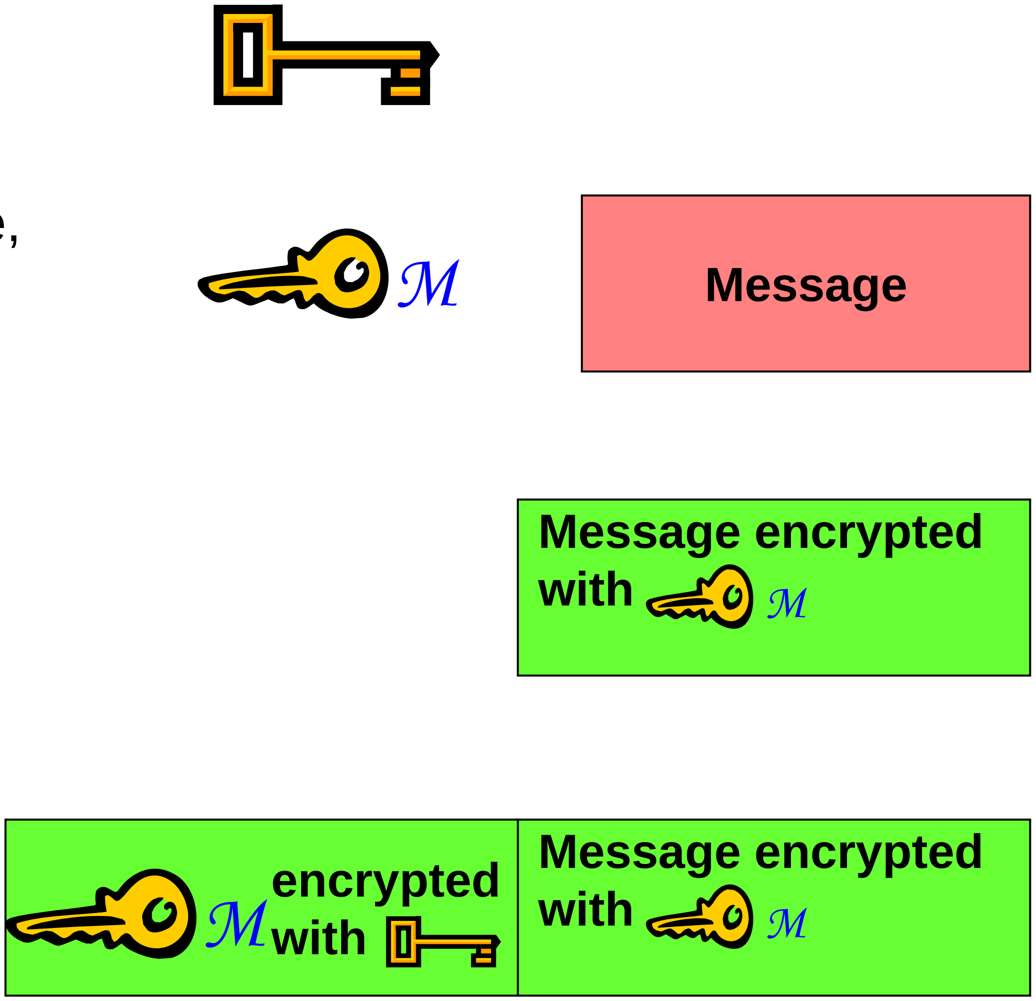
Encrypting a Message

Each member has a neighborhood key.

When member sends a message, it creates a message key for this message.

Then, it encrypts the message with message key.

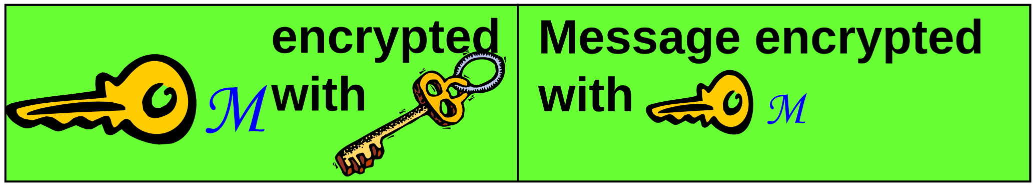
Finally, ite encrypts the message key with its neighborhood key and adds it to the message.



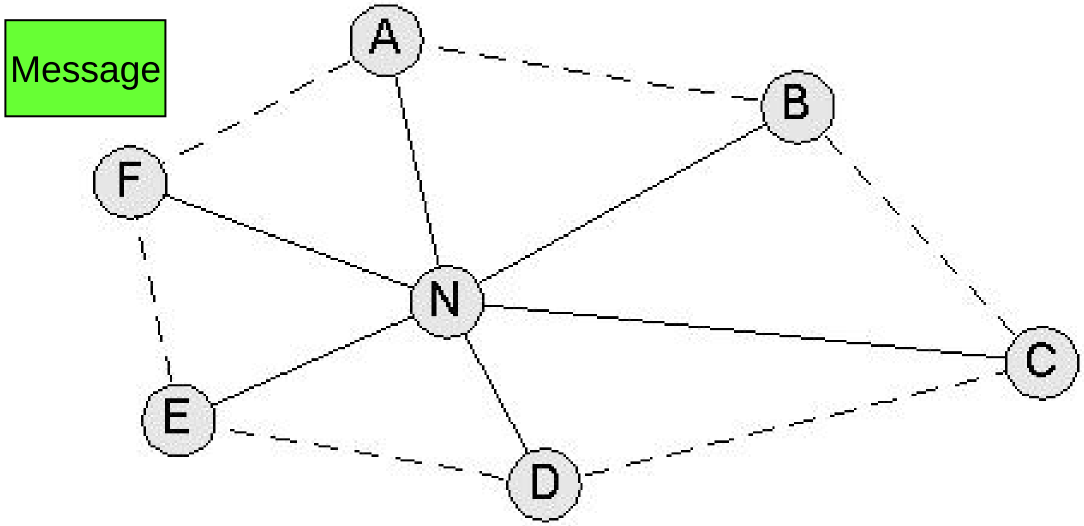
$\mathcal{M}$
Message
Message encrypted with $\mathcal{M}$
$\mathcal{M}$ encrypted with
Message encrypted with $\mathcal{M}$

Forwarding an Encrypted Message

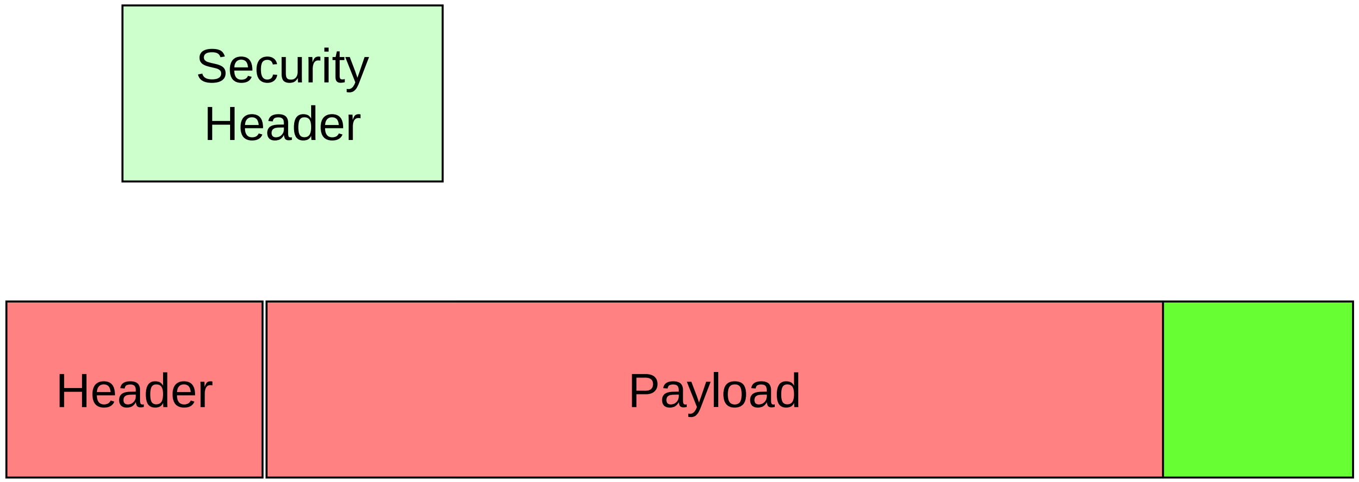
N re-encrypts the message key with its personal key



$\mathcal{M}$ encrypted with
Message encrypted with $\mathcal{M}$



Processing of Encrypted Message

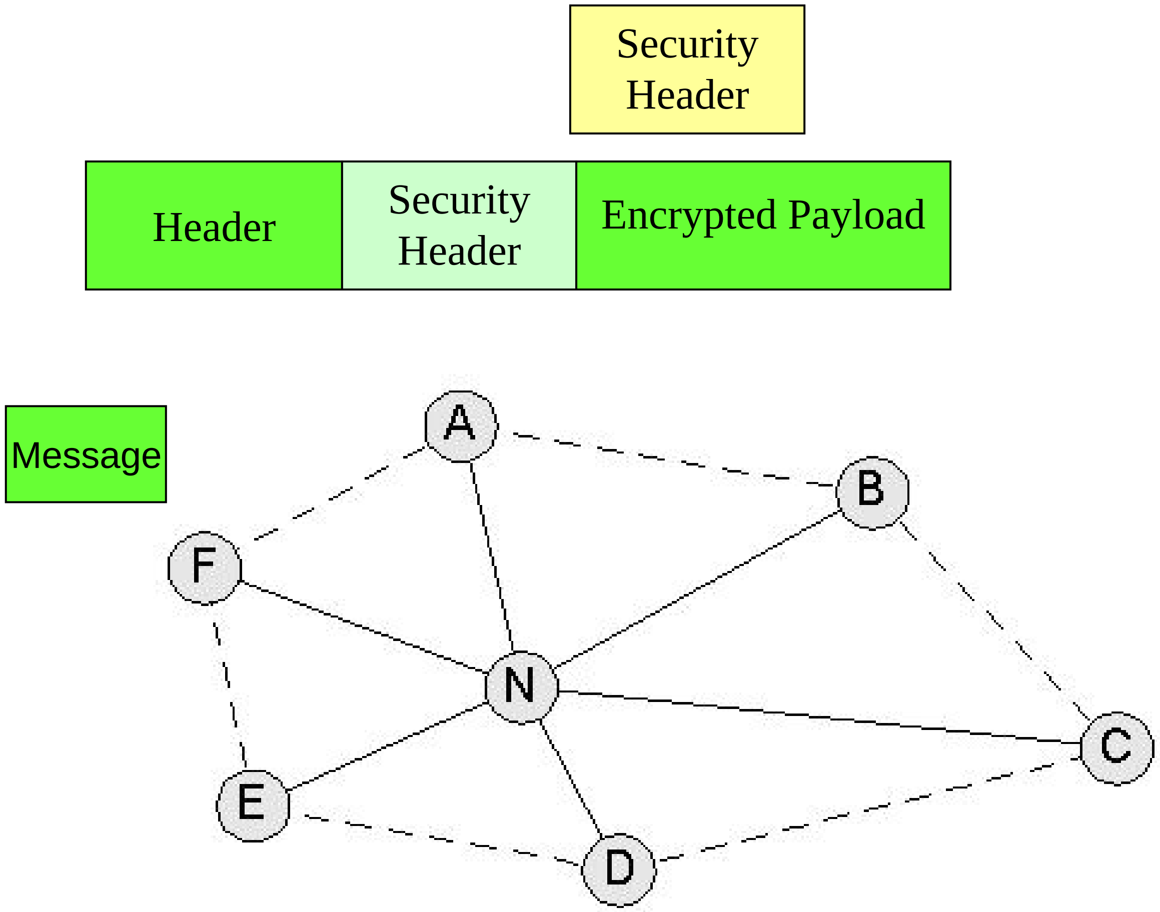


- Keys for header and payload are kept in a security header
- Permits separation of security for payload and header

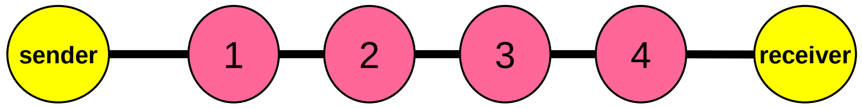
Processing of Encrypted Message

- Neighborhood key scheme amounts to exchanging security header at intermediate nodes

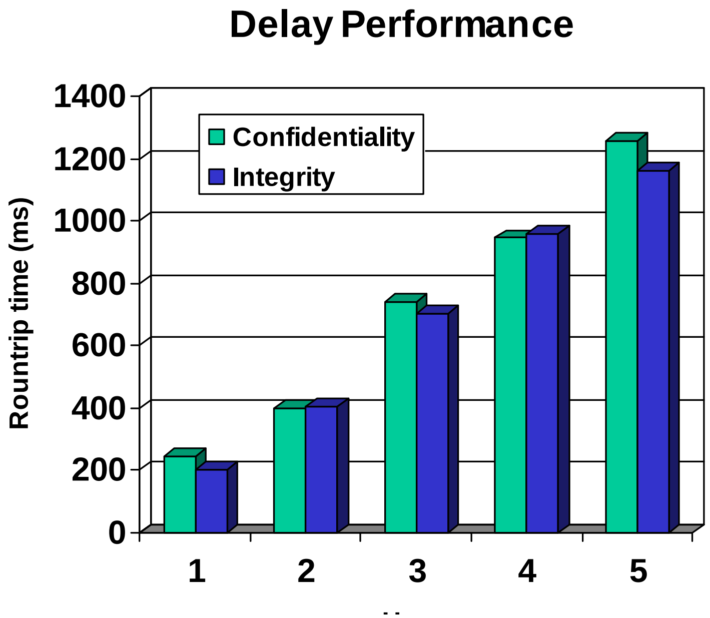
- *Note: Node sends same message to **all** neighbors !*

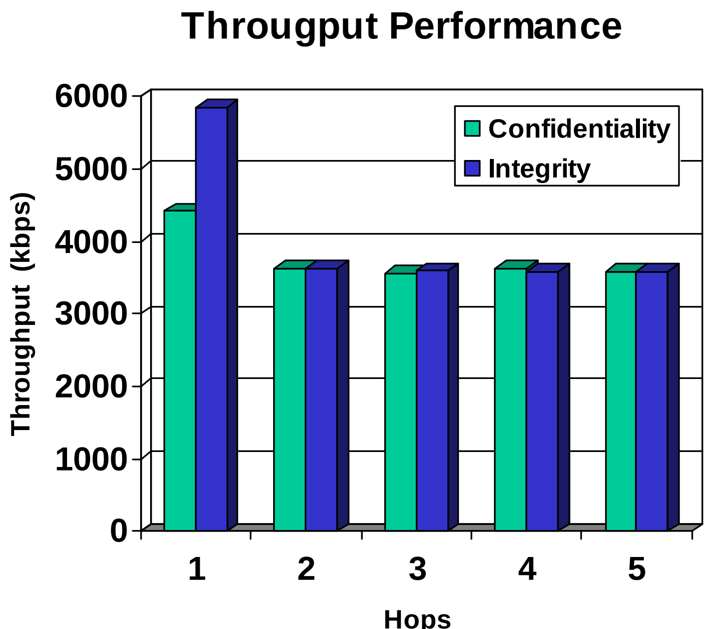


Experiment



- Overlay consists of sender, receiver and 4 intermediate peers
- Each peer is running on a separate PC
- Overlay uses TCP between peers
- Sender transmits 10,000 messages of a given size to receiver and receiver sends small acknowledgements
- Message size is 2048 bytes
- Sender records time when message returns





Monitoring and control of groups

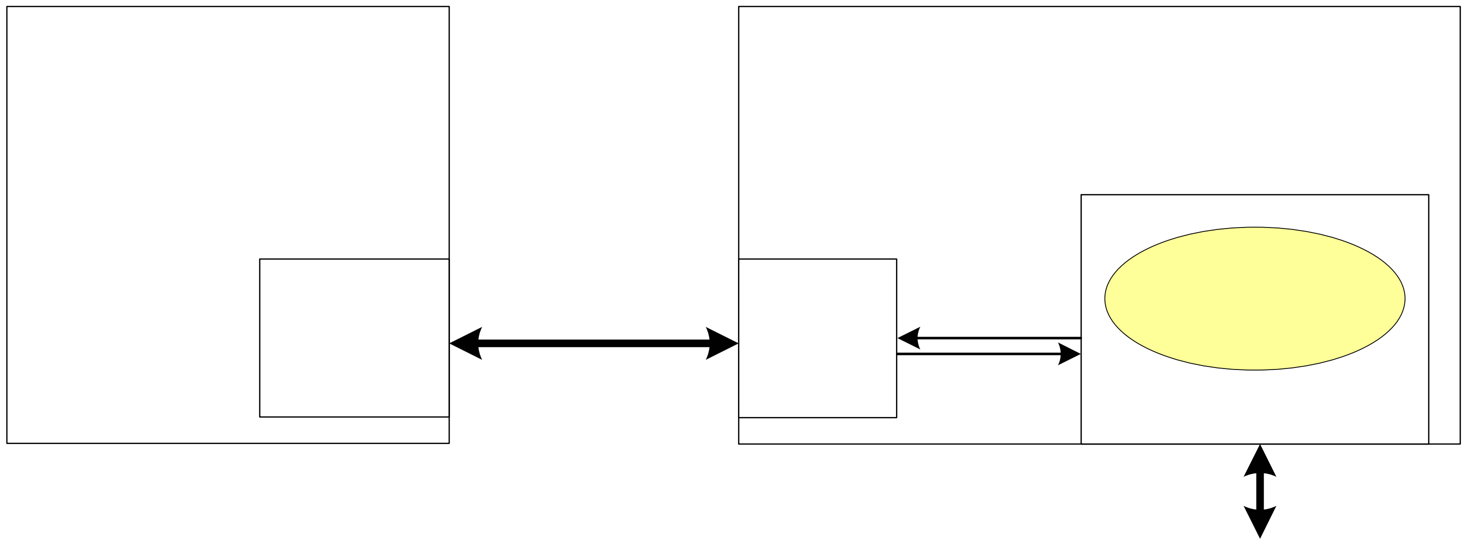
Monitor and Control System

- **Loosely modeled after SNMP:**
  - Each application component collects statistics
  - Statistics can be accessed by a remote monitor

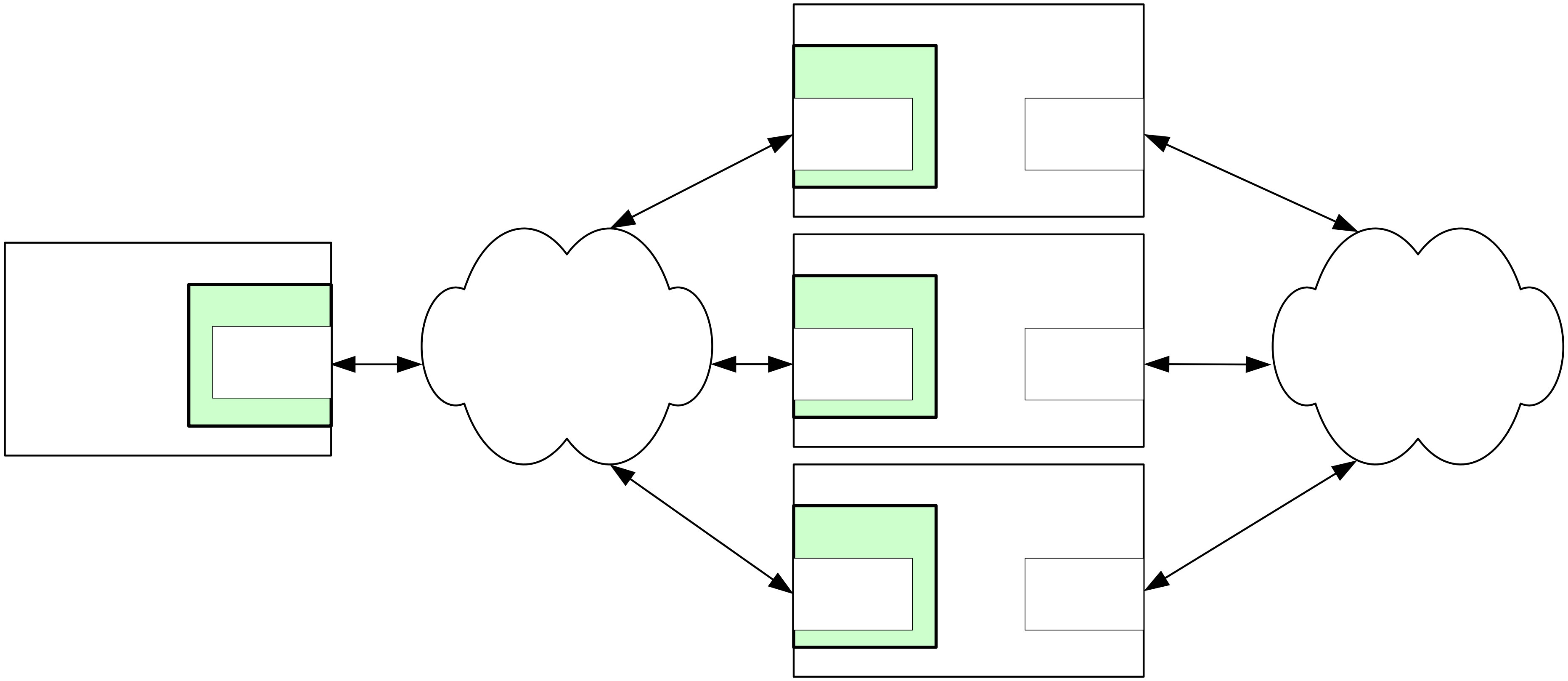
- **XML oriented:**
  - Statistics are internally stored as XML documents
  - Transmitted messages have XML format

- **Dynamically created content:**
  - Structure of XML documents with statistics is created dynamically upon receiving a query
  - Application can add statistics to an application program

Monitors and Portals



Monitor Overlay Network



Hierarchy of statistics

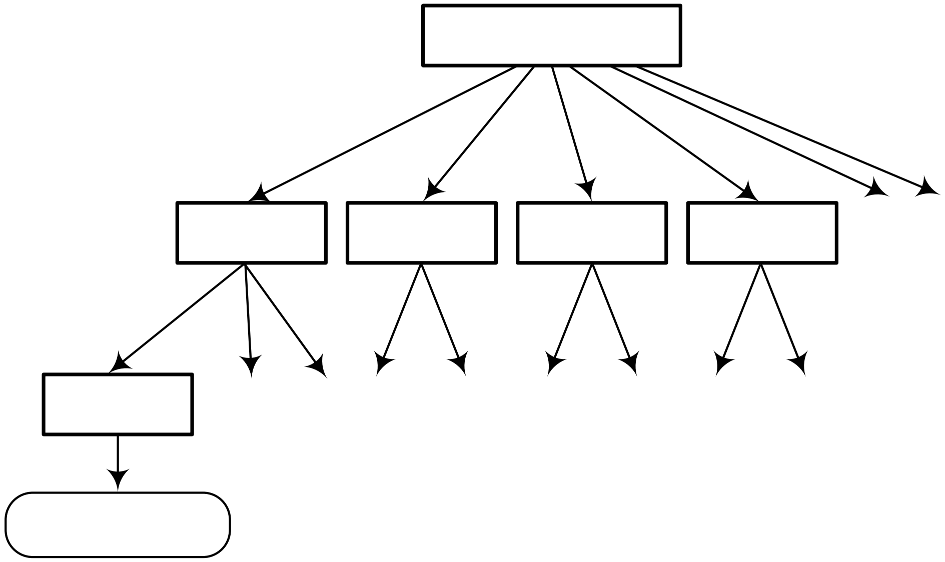
```
<Socket>
    <Node>
        ….
        <NodeAdapter>
            ….
            <UBytesSent> 1004 </UBytesSent >
        </NodeAdapter>
    </Node>

    <Config> …. </Config >

    <RecvBuf> …. </RecvBuf>

    <SocketAdapter> ….  </SocketAdapter>

</Socket>
```



Accessing Statistics

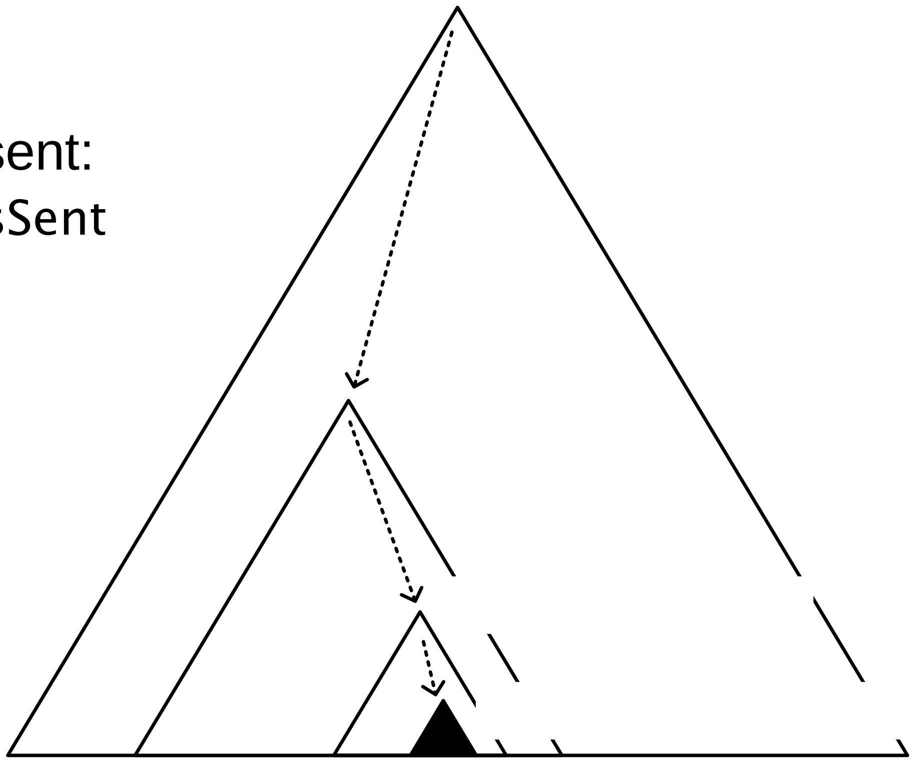
- Statistics are accessed using XPath expressions

- Adressing the number of bytes sent:

/Socket/Node/NodeAdapter/UBytesSent

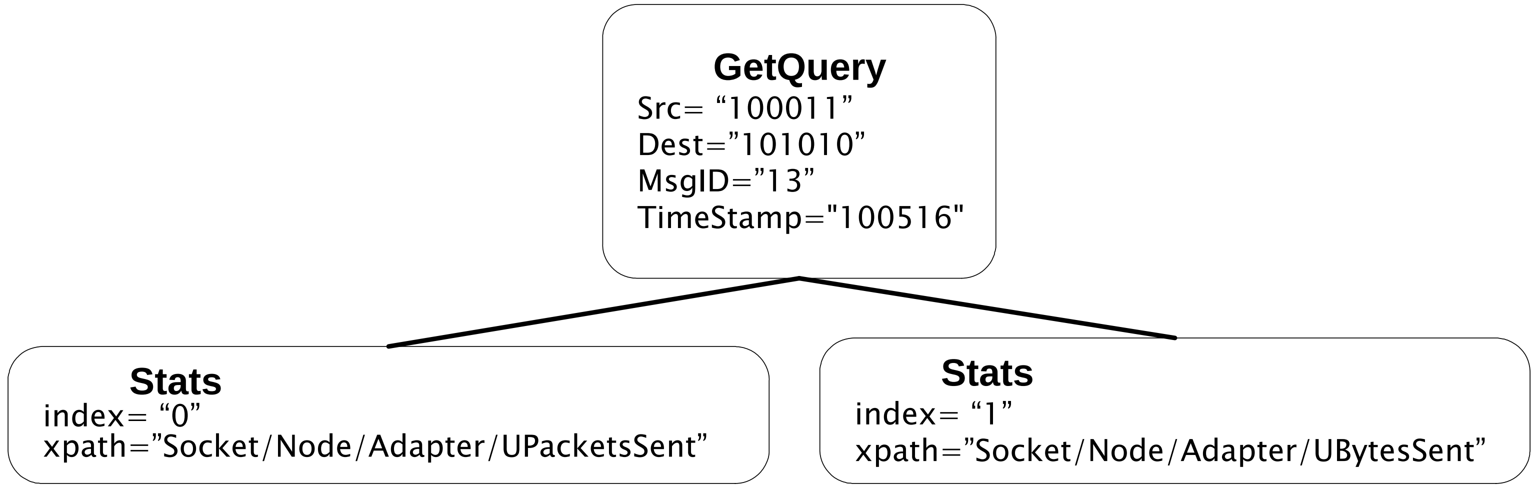
- Addressing all statistics of the overlay node:

/Socket/Node



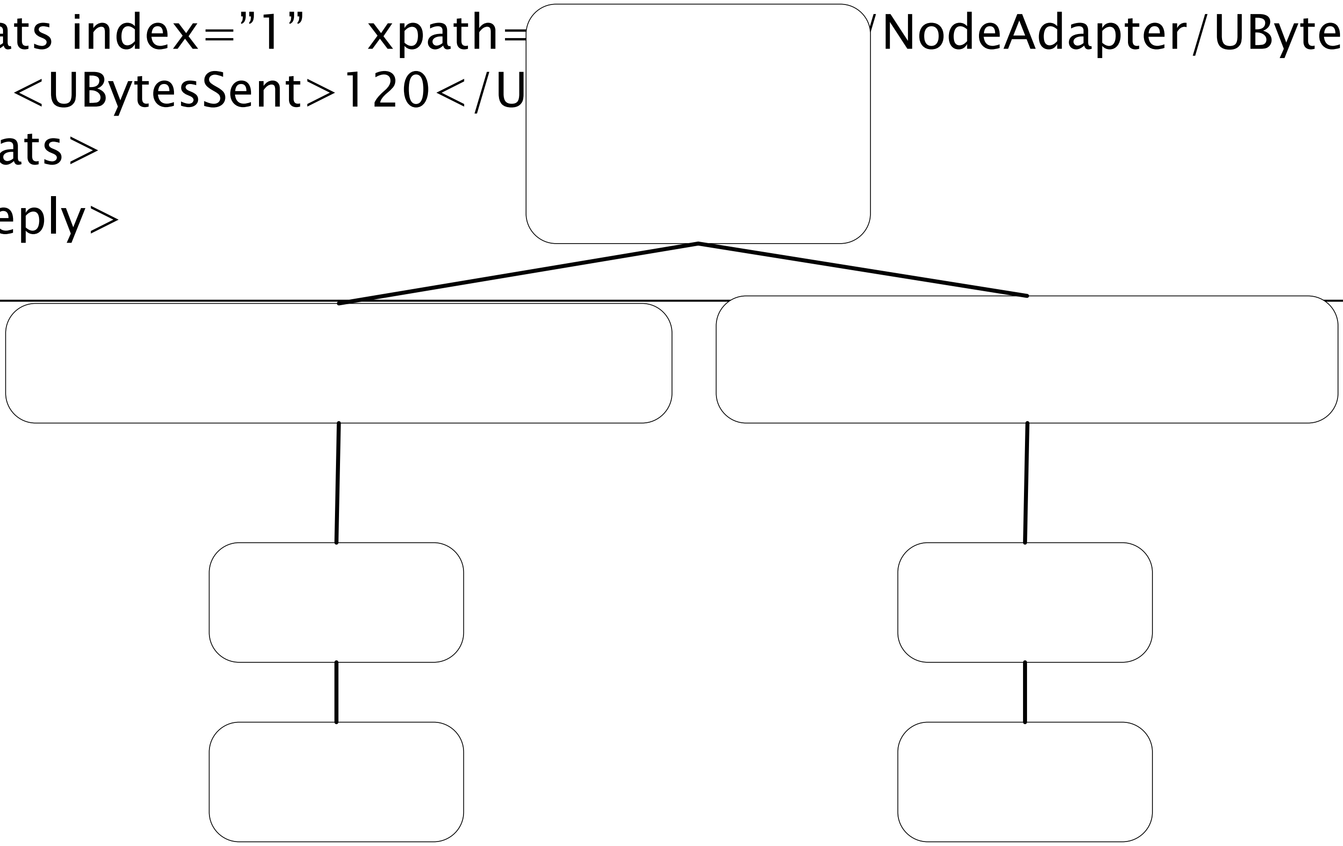
Query for statistics

```
<GetQuery        Src= “100011”  Dest=”101010”  MsgID=”13”
                 TimeStamp="100516">
   <Stats index=”0”
                 xpath=”/Socket/Node/NodeAdapter/UPacketsSent” />
   < Stats index=”1”
                 xpath=”/Socket/Node/NodeAdapter/UBytesSent” />
</GetQuery>
```

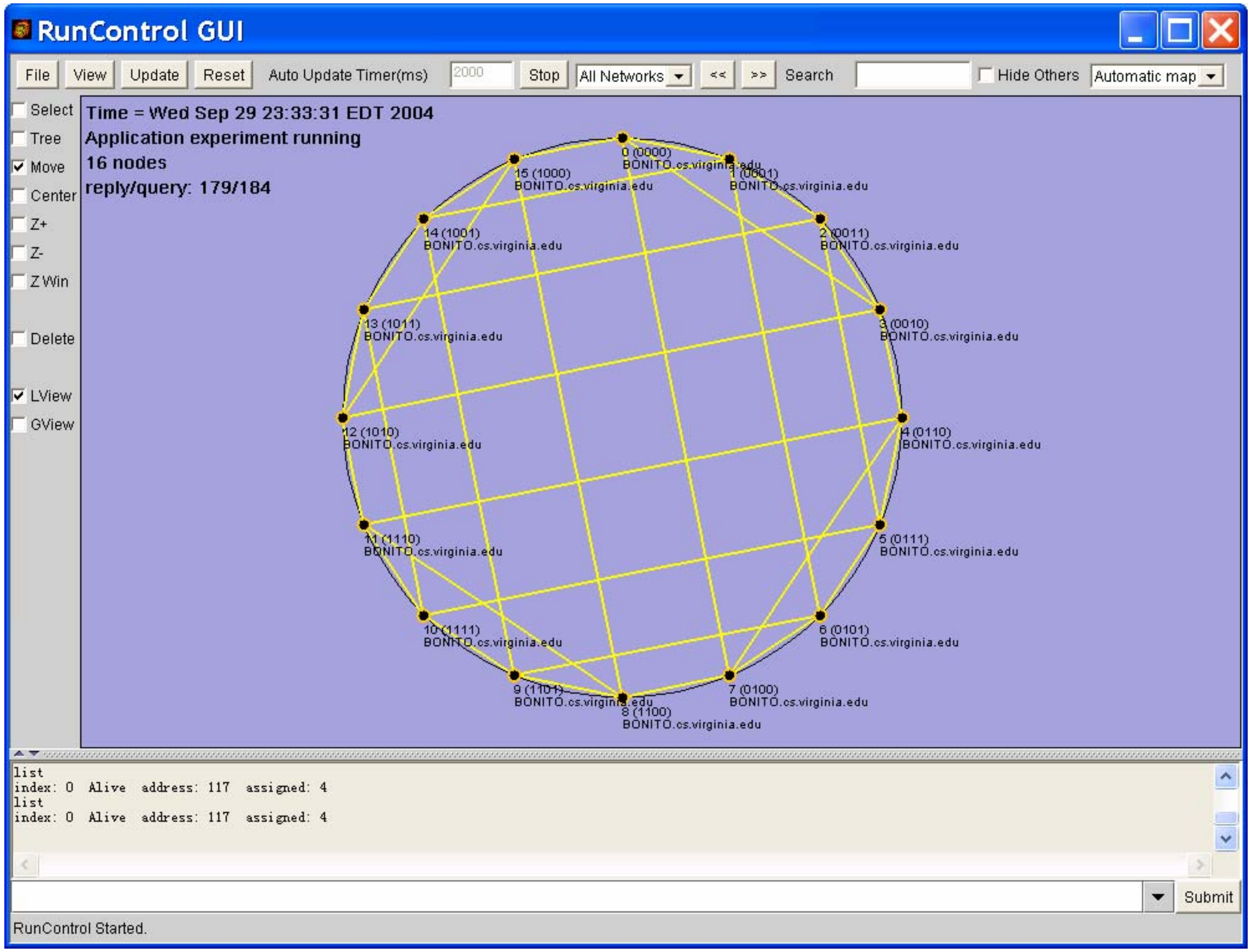


Response to query

```
<GetReply        Src= “101010”  Dest=”100011”  MsgID=”13”
          TimeStamp="106340">
    <Stats index=”0” xpath=”/Socket/Node/NodeAdapter/UPacketsSent” >
          <UPacketsSent>120</UPacketsSent>
    </Stats>
    < Stats index=”1”    xpath=                /NodeAdapter/UBytesSent”>
          <UBytesSent>120</U
    </Stats>
</GetReply>
```



GUI for monitoring an Overlay Network


RunControl GUI
File
View
Update
Reset
Auto Update Timer(ms)
2000
Stop
All Networks
<<
>>
Search
Hide Others
Automatic map
Select
Tree
Move
Center
Z+
Z-
Z Win
Delete
LView
GView
0 (0000) BONITO.cs.virginia.edu
1 (0001) BONITO.cs.virginia.edu
2 (0011) BONITO.cs.virginia.edu
3 (0010) BONITO.cs.virginia.edu
4 (0110) BONITO.cs.virginia.edu
5 (0111) BONITO.cs.virginia.edu
6 (0101) BONITO.cs.virginia.edu
7 (0100) BONITO.cs.virginia.edu
8 (1100) BONITO.cs.virginia.edu
9 (1101) BONITO.cs.virginia.edu
10 (1111) BONITO.cs.virginia.edu
11 (1110) BONITO.cs.virginia.edu
12 (1010) BONITO.cs.virginia.edu
13 (1011) BONITO.cs.virginia.edu
14 (1001) BONITO.cs.virginia.edu
15 (1000) BONITO.cs.virginia.edu
Submit

Time = Wed Sep 29 23:33:31 EDT 2004
Application experiment running
16 nodes
reply/query: 179/184

```
list
index: 0  Alive  address: 117  assigned: 4
list
index: 0  Alive  address: 117  assigned: 4
```

RunControl Started.

More Information

HyperCast web site:

http://hypercast.org

Design documents, download software, user manual

- Send questions to **hypercast@cs.virginia.edu**
- Downloadable software is from 2002 (Version 2.0)
- Version 3.0 with security, message semantics, ad-hoc support, management and control is in development