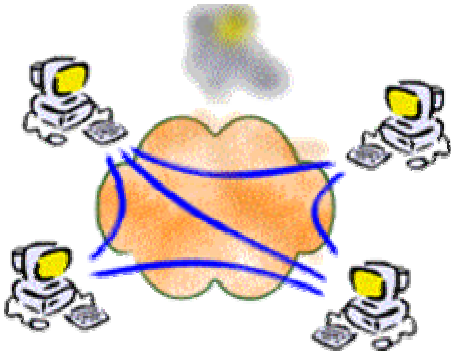
Two challenges for building large self-organizing overlay networks

Jorg Liebeherr

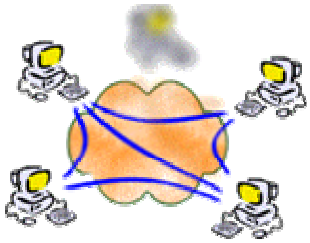
University of Virginia



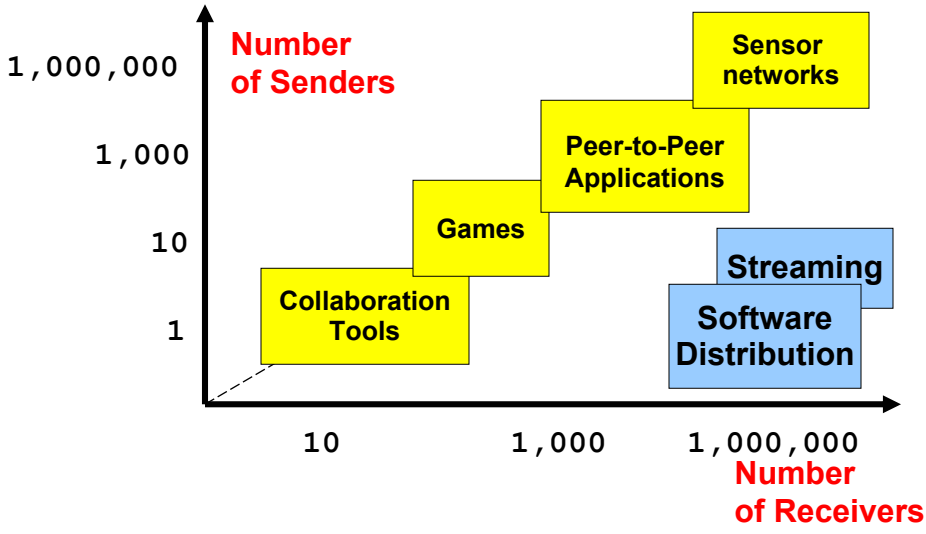
Two issues in multicast overlay networks

1. **Why do we keep on proposing overlay networks for multicast?**

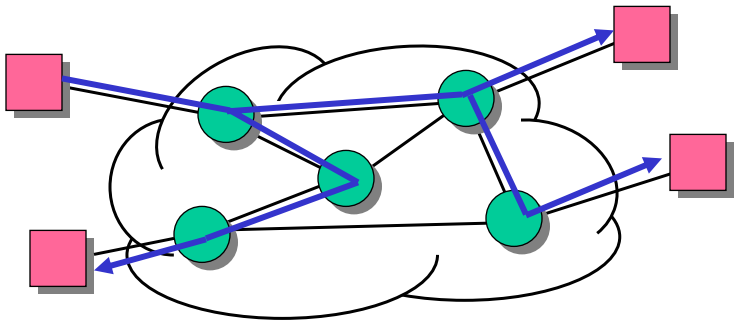
2. **Why is it difficult to write application programs for overlay network?**



Applications with many receivers

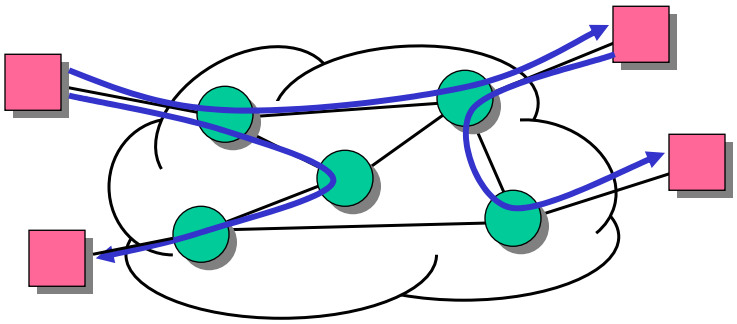


Multicast support in the network infrastructure (IP Multicast)



- **IP Multicast problems:**
  - 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



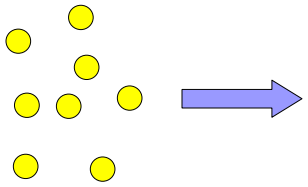
- **Logical overlay** resides on top of the Layer-3 network
- Data is transmitted between neighbors in the overlay
- No network support needed
- Overlay topology should match the Layer-3 infrastructure

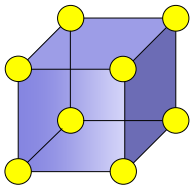
Overlay-based approaches for multicasting

- 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 distributed hash tables (DHT) and embed trees:
  - Chord (UCB, MIT)
  - CAN (UCB, ICIR)
  - Pastry (Rice, MSR)
  - many more

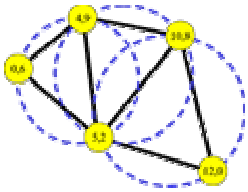
Own Approach

- **Build overlay network as a graph with known properties**
  - N-dimensional (incomplete) hypercube *(1997-1999)*
  - Delaunay triangulation *(1999-2002)*
  - Spanning tree





hypercube



Delaunay triangulation

What is the best overlay?

Evaluation criteria:

1. **Properties of the overlay graph**
2. **Mapping of the overlay to the layer-3 network**
3. **Properties of protocol that maintains the overlay topology**

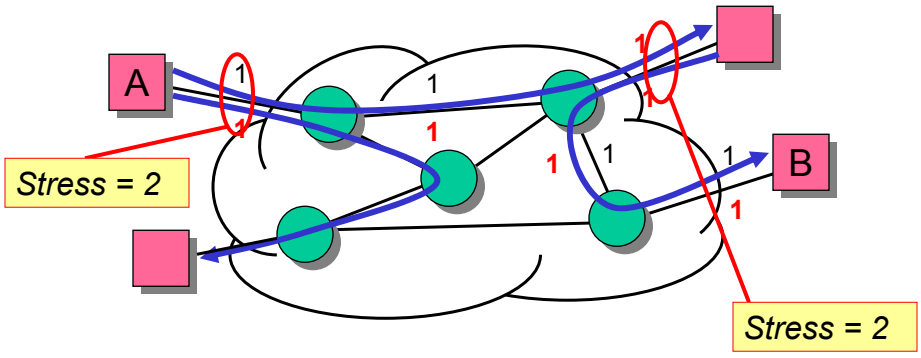
1. Properties of the overlay graph

- **Number of neighbors (routing table size)**
  - **Many DHTs, hypercubes: O(log N)** *(max.)*
  - **Triangulation graphs: O(N)** *(max.),* **6** *(avg.)*
  - **Meshes, trees:** no *a priori* bound, but bounds can be enforced
- **Path lengths in the overlay**
  - **Many DHTs, hypercubes: O(log N)** *(max.)*
  - **Triangulation graphs: O(N)** *(max.),* **O(√N)** *(best case avg.)*
  - **Meshes, trees:** no *a priori* bound

2. Mapping of the overlay to the layer-3 network

- **Compare overlay multicast to network-layer multicast:**
  "Relative Delay Penalty": Ratio of delay to shortest path delay
  "Stress": Number of duplicate transmissions over a physical link
- Overlays that provide a good mapping to need to be aware of the underlying layer-3 network

Illustration of “Stress” and “Relative Delay Penalty”



Unicast delay A→B : 4

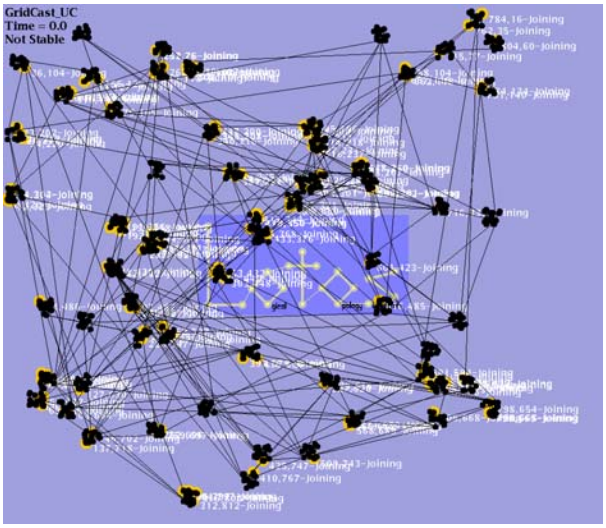
Delay A→B in overlay: 6

Relative delay penalty for A→B: 1.5

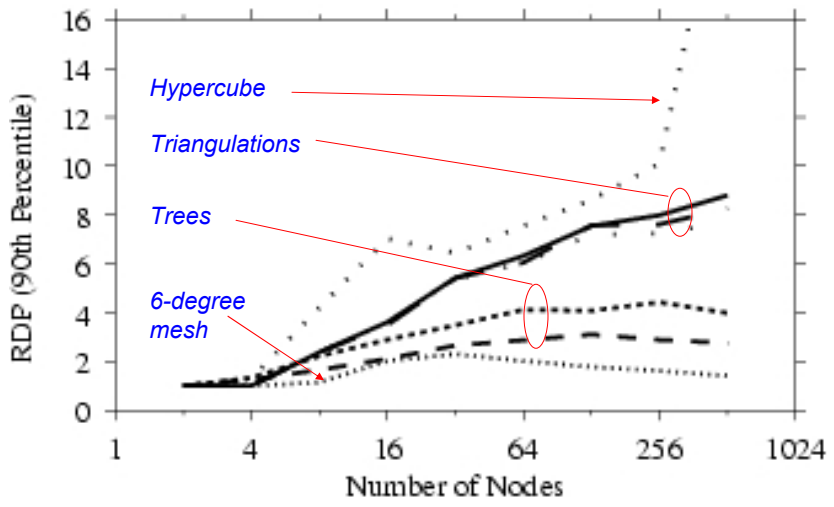
Transit-Stub Network

Transit-Stub

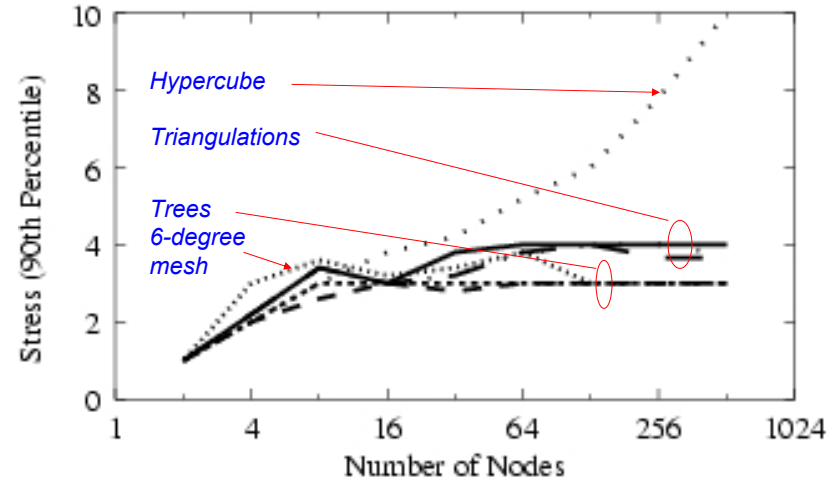
- GA Tech topology generator
- 4 transit domains
- 4×16 stub domains
- 1024 total routers
- 128 hosts on stub domain



90th Percentile of Relative Delay Penalty



90th Percentile of "Stress"



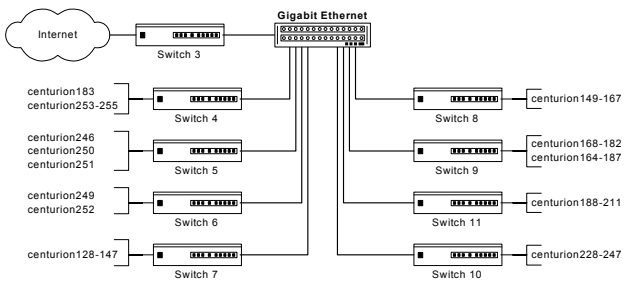
3. Properties of overlay protocol

- **Measures:**
  - How fast does a self-organizing protocol converge?
  - How does protocol behavior change when
    - … the size of overlay network grows (scalability)?
    - …. the multicast group is highly dynamic?

- **Example: Delaunay Triangulation protocol**

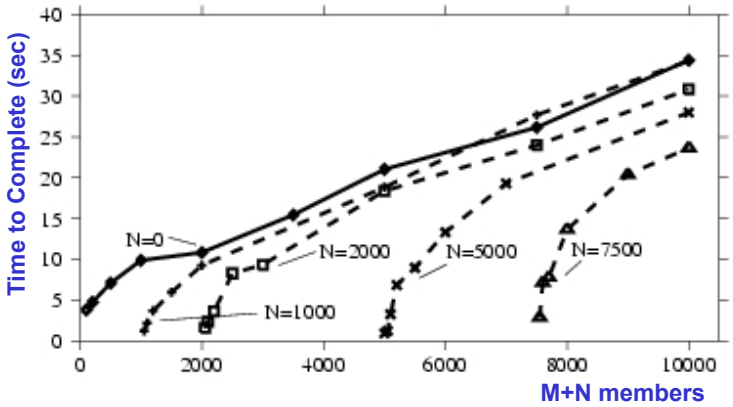
Measurement Experiments

- **Experimental Platform:**
  Centurion cluster at UVA (cluster of 300 Linux PCs)
  - **2 to 10,000 overlay members**
  - **1–100 members per PC**
- **Overlay topology: Delaunay triangulation with random coordinate assignments**



Experiment with Delaunay Triangulation: Time to complete an overlay network

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



Economy-of-scale versus increased scalability

Best use of network resources requires good mapping of overlay network to layer-3 network

(which in turn requires measurements or awareness of layer-3 network)

- → **Interferes with the ability of building overlay graphs with good a priori bounds**
- → **Generally, increases convergence times of the protocols**
- → **Limits the ability of a protocol to support very large groups**

Overlay networks that have good *a priori* bounds and overlay protocols that scale to very large groups generally ignore the layer-3 network

- → **can lead to a poor match to the layer-3 network**
- → **can lead to poor use of resources**

- *No best solution, but a trade-off*
- ***Achieve scalability by trading off economy of scale***
- *Design space given by this trade-off is still not well understood*

Programming overlay networks

- Research focuses on the design of protocols to maintain and forward data in an overlay network
- Less attention is put on building application programs in such an environment

- ***Overlay network programming*** *is* the software development process of building application programs that communicate with one another in an application-layer overlay network

Application programming interfaces (APIs) for overlay networks

- Many overlay network protocols do not shield API from overlay network protocol

- **Notable exceptions:**
  - Socket-based API: Yoid, Scattercast
  - API for DHT overlays: F. Dabek et. al. (IPTPS 03)
  - Rendezvous based abstractions: I3 (by Stoica et. al.)

- Also:
  - JXTA: abstractions for P2P applications

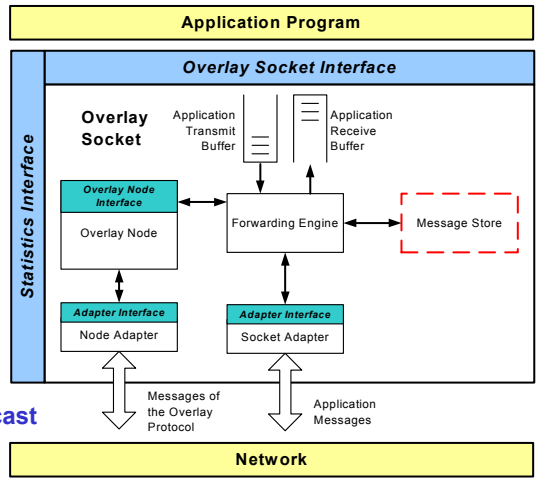
Our work: Overlay Sockets

An **overlay socket** provides a socket-based API:

1. Does not require knowledge of the overlay network topology
2. Accommodates different overlay network topologies
3. Accommodates different types of transport layer protocols (TCP, UDP, UDP multicast)
4. Provides mechanisms for bootstrapping new overlay networks

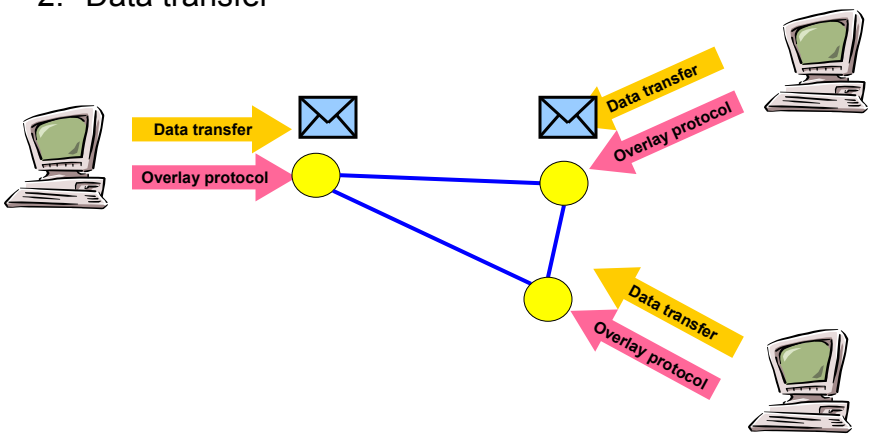
Structure of an Overlay Socket

- Transport services in peer networks
- Socket-based API
- UDP, TCP, UDP multicast
- Different transport service semantics
- Implementation in Java
- Software available from: **www.cs.virginia.edu/hypercast**



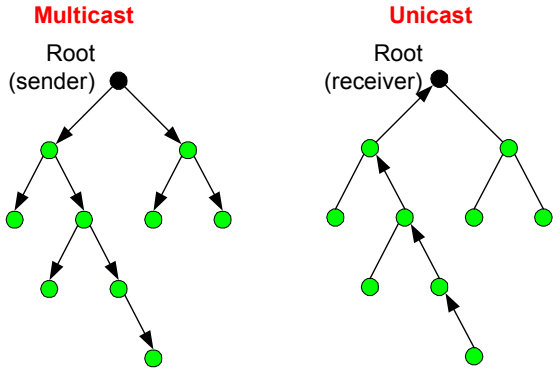
Overlay Socket: Data Exchange

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



Unicast and Multicast in overlays

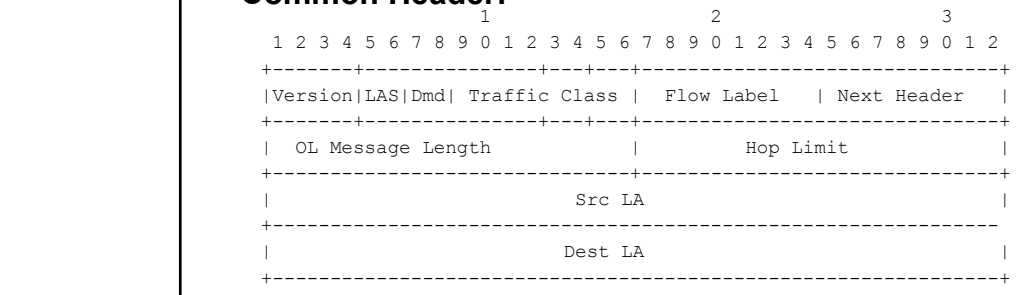
- Unicast and multicast is done using trees that are embedded in the overlay network.
- Requirement: Overlay node must be able to compute the child nodes and parent node with respect to a given root



Overlay Message Format

Loosely modeled after IPv6 → minimal header with extensions

- **Common Header:**



Version (4 bit):	Version of the protocol (current Version is 0x0)
LAS (2 bit):	Size of logical address field
Dmd (4bit)	Delivery Mode (Multicast, Flood, Unicast, Anycast)
Traffic Class (8 bit):	Specifies Quality of Service class (default: 0x00)
Flow Label (8 bit):	Flow identifier
Next Header (8 bit)	Specifies the type of the next header following this header
OL Message Length (8 bit)	Specifies the type of the next header following this header.
Hop Limit (16 bit):	TTL field
Src LA ((LAS+1)*4 bytes)	Logical address of the source
Dest LA ((LAS+1)*4 bytes	Logical address of the destination

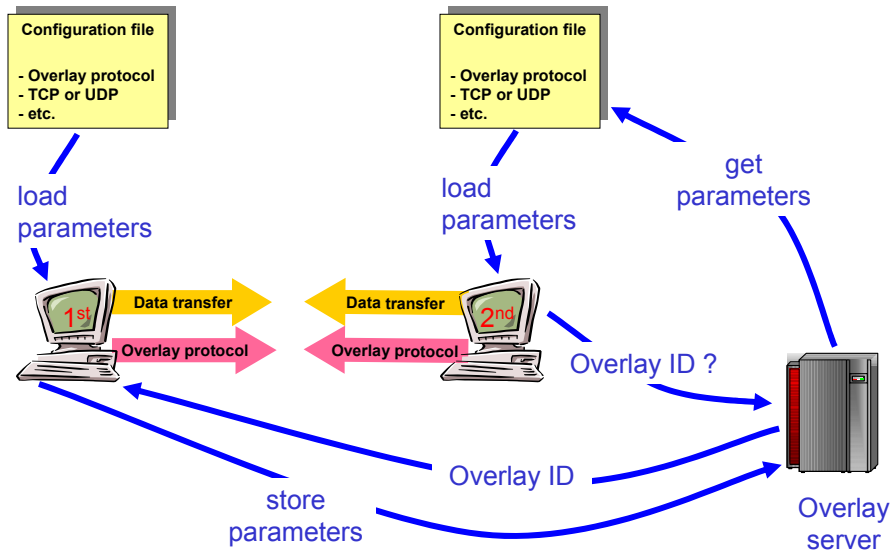
Property File

- Stores attributes that configure the overlay socket (overlay protocol, transport protocol and addresses)

```
# This is the Hypercast Configuration File
#
#
# LOG FILE:
LogFileName = stderr

# ERROR FILE:
ErrorFileName = stderr

# OVERLAY Server:
OverlayServer =

# OVERLAY ID:
OverlayID = 224.228.19.78/9472
KeyAttributes = Socket,Node,SocketAdapter

# SOCKET:
Socket = HCast2-0
HCAST2-0.TTL = 255
HCAST2-0.ReceiveBufferSize = 200
HCAST2-0.ReadTimeout = 0

...
```

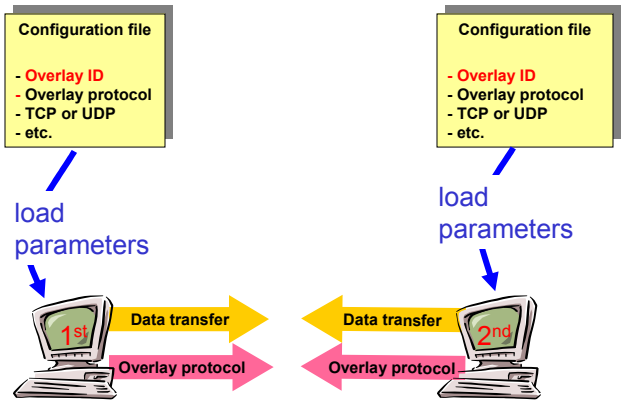
```
# SOCKET ADAPTER:
SocketAdapter = TCP
SocketAdapter.TCP.MaximumPacketLength = 16384
SocketAdapter.UDP.MessageBufferSize = 100

# NODE:
Node = HC2-0
HC2-0.SleepTime = 400
HC2-0.MaxAge = 5
HC2-0.MaxMissingNeighbor = 10
HC2-0.MaxSuppressJoinBeacon = 3


# NODE ADAPTER:
#
NodeAdapter = UDPMulticast

NodeAdapter.UDP.MaximumPacketLength = 8192
NodeAdapter.UDP.MessageBufferSize = 18
NodeAdapter.UDPServer.UdpServer0 = 128.143.71.50:8081
NodeAdapter.UDPServer.MaxTransmissionTime = 1000
NodeAdapter.UDPMulticastAddress = 224.242.224.243/2424
```

Overlay socket: Bootstrap



Overlay socket : Bootstrap

(without Overlay server)



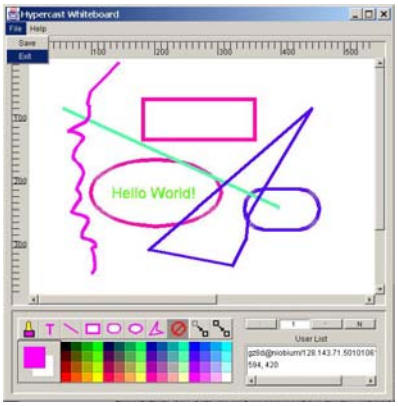
Socket Based API

- Tries to stay close to Socket API for UDP Multicast
- Program is independent of overlay topology

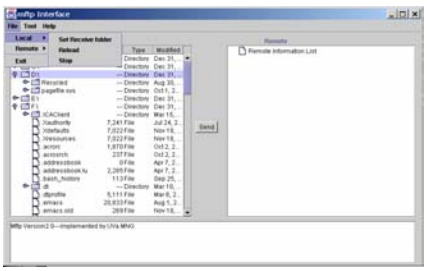
```
//Generate the configuration object
OverlayManager om = new OverlayManager("hypercast.prop");
String overlayID = om.getDefaultProperty("MyOverlayID")
OverlaySocketConfig config = new
om.getOverlaySocketConfig(overlayID);

//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

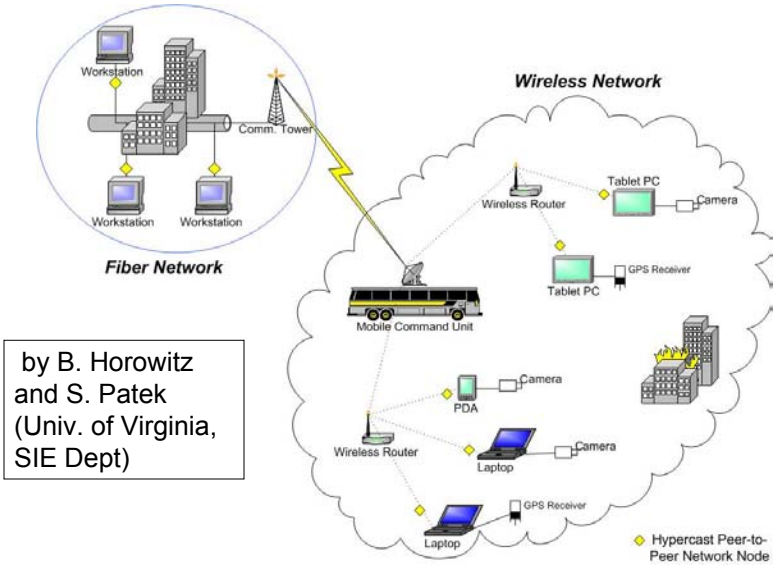


Multicast file transfer



Data aggregation in P2P Net: → CS757 Homework

More advanced application: Emergency Response Network



by B. Horowitz and S. Patek (Univ. of Virginia, SIE Dept)