# OBAMP
# Overlay, Borůvka-based, Ad-hoc Multicast Protocol

Andrea Detti, Nicola Blefari-Melazzi, Remo Pomposini, Saverio Proto

University of Rome "Tor Vergata", Electronic Engineering Dept., Italy

{andrea.detti,nicola.blefari} @ uniroma2.it

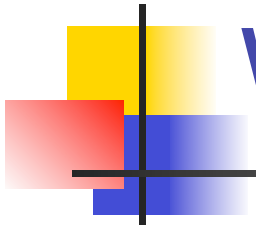
remo.pomposini@gmail.com

zioproto@gmail.com

http://www.radiolabs.it/obamp

# What is OBAMP ?

- MANET overlay multicast protocol, i.e. peer-to-peer protocol
  - user data distributed over a shared distribution tree formed by UDP tunnels among member nodes
  - only member nodes perform the multicast routing
  - developed at application layer
- OBAMP shows three distinctive advantages:
  - builds an efficient distribution tree
  - exploits radio broadcasting
  - limits the overall signaling load, network+overlay
- As a consequence, OBAMP has a low-latency and a high delivery ratio, even when the group size increases
- Feasibility proof through JAVA tested

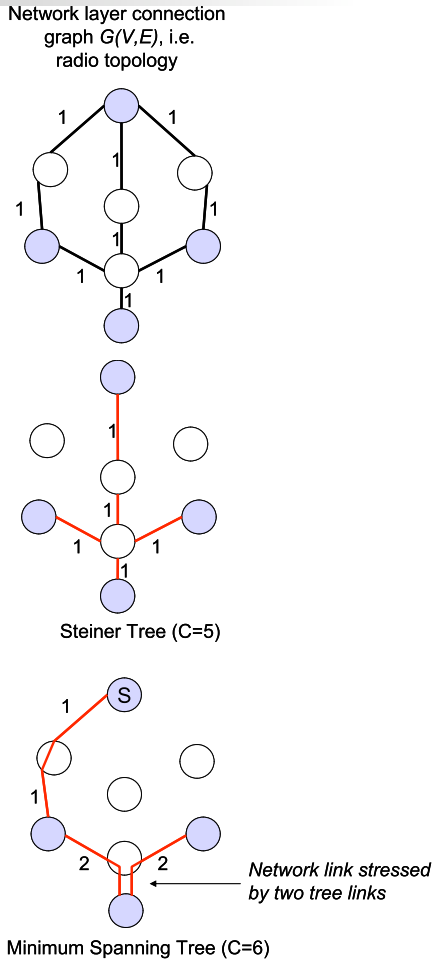


# Why we need OBAMP?

- **Multicast** Radio streaming over Manet
- **Multicast** TV streaming over Manet
- Other applications??

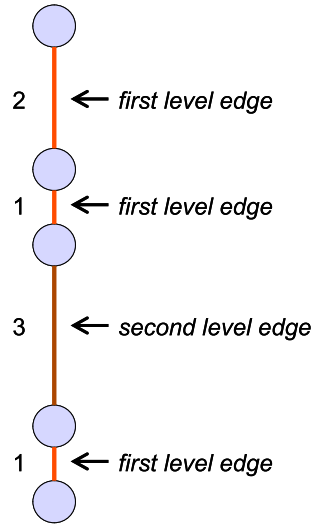
# Graph theory background 1/2

- What is the cheapest distribution tree ?
  - the minimum spanning tree for the overlay approach
  - the Steiner tree for the network layer approach
- The overlay approach can not exploit non-member nodes for multicast routing; this implies that some network links may be stressed by more than one transmission of the same data. For this reason the overlay approach is less efficient than the network-layer approach (i.e., Steiner tree cheaper than minimum spanning tree)
  - The average ratio between Steiner and minimum spanning tree cost is limited to 0.9
- This limited efficiency penalty makes attractive the overlay approach, if the relevant deployment simplification of an application layer protocol is take into account



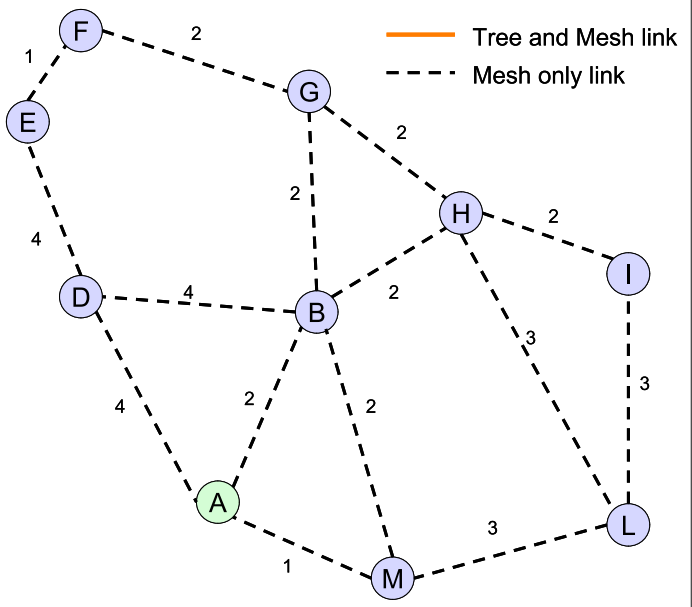
# Graph theory background 2/2

- Borůvka algorithm overview (1926)
  - finds the minimum spanning tree over a given graph, such as Kruskal (1956) and PRIM algorithm (1957)
  - it was proposed for electrical networks
  - it lends itself more easily to a distributed implementation
- The algorithm
  - `make a list L of {W} trees, where each tree is composed of a single vertex`
  - `while L has more than one tree`
    - `for each tree in L, find the smallest edge connecting the tree to another disjoined tree, thus forming a new tree`
  - `end`
- The graph vertex is the member node; the graph edge is the transport connection, which cost is the number of underling network hops
- Let us define **first-level edges** the set of edges of the minimum spanning tree built by the Borůvka algorithm at the first iteration (in the while loop). The first-level edges are the edges that connect nearest members



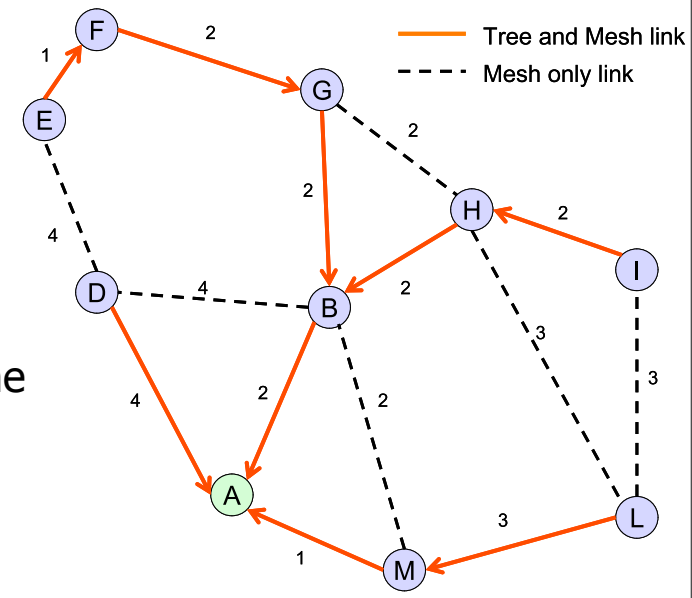
# Basic operations of OBAMP: mesh-create

- OBAMP follows a mesh-first approach
  - it maintains an overlay network among member nodes named mesh
  - within the mesh, it selects the overlay links that forms the distribution tree
- The mesh is periodically refreshed through an expanding-ring search approach, based on HELLO messages (note: only on AODV mesh -> flooding)
- The mesh links are maintained by member nodes in a soft-state approach: those not refreshed are pruned
- Mesh property: **the mesh surely contains the first-level edges of the minimum spanning tree**



# Basic operations of OBAMP: tree-create

- Upon the mesh, OBAMP builds an hard-state shared distribution tree
  - hard-state: the release and the setup of a tree links are regulated by specific two way handshake procedures
- The tree is periodically refreshed by the core member, which floods the mesh with TREECREATE messages
- During the flooding, each member selects its current closest upstream member of the tree toward the core, setups the relevant tree link and, in case, tears down the tree link toward the old upstream member
  - the selection procedure consists in choosing as closest upstream member, the member that has sent the first "handled" TREECREATE
    - between the reception and the handling, the receiving member applies an **`HANDLING_DELAY`** computed as follows: 0 if the TREECREATE comes from the nearest member; $\Delta$ * "sender hop distance", otherwise

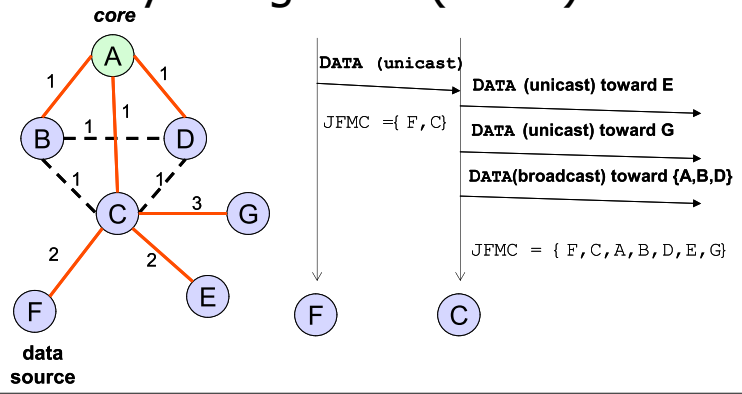



*Arrow indicates the closest upstream member*

- Tree property: the resulting tree at least contains the first-level edges of the minimum spanning tree. **OBAMP tree is an approximation of the minimum spanning tree**

# Basic operations of OBAMP: data-distribution

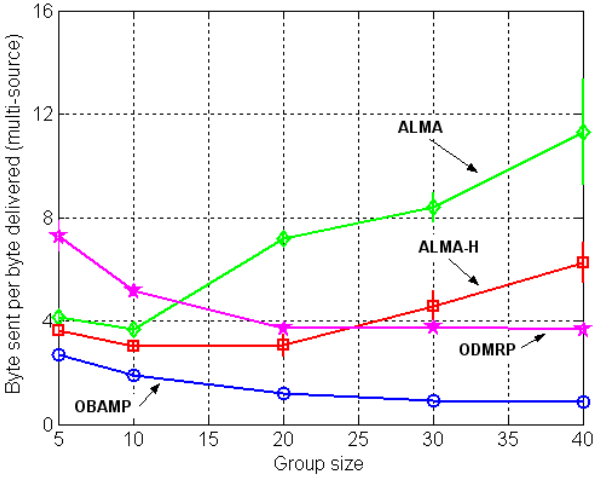
- Data distribution is performed through both unicast and broadcast UDP tunnels
  - At the reception of a data packet from F, the member C forwards this data toward all neighbours which distance is greater than one hop and which are connected by a tree link (i.e., E and G). Each forwarding is carried out by the transmission of an unicast UDP/IP packet.
  - Afterward, the member C forwards the received data toward all neighbours which distance is equal to one hop (i.e., A, B and D). This latter forwarding is carried out by a single transmission of a broadcast UDP/IP packet with IP TTL=1
- To limit data duplication each data message contains the list of the members to which those data have already being sent. (`JFMC`)

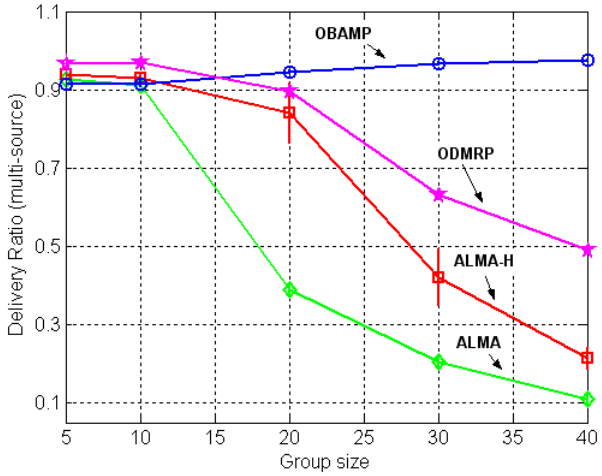


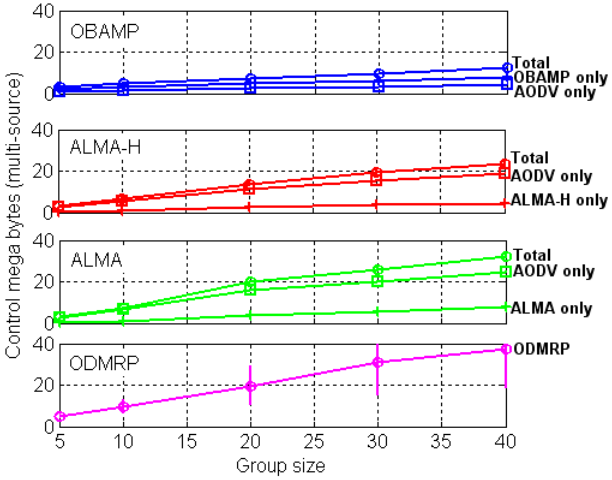
# Performance evaluation

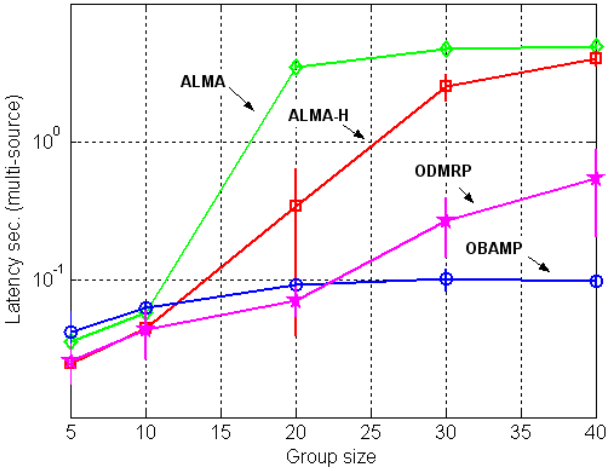
- NS2
- RANDOM WAY POINT (speed 10m/s, pause 30s)
- 95 % confidence interval (10 sims run of 800s)
- Area 1000x1000
- 50 nodes
- 802.11 @ 2Mbps with 250 m of coverage
- 16 kbps total CBR offered traffic
  - One-to-all (i.e., single source)
  - All-to-all (i.e., multi source)

# Multi source

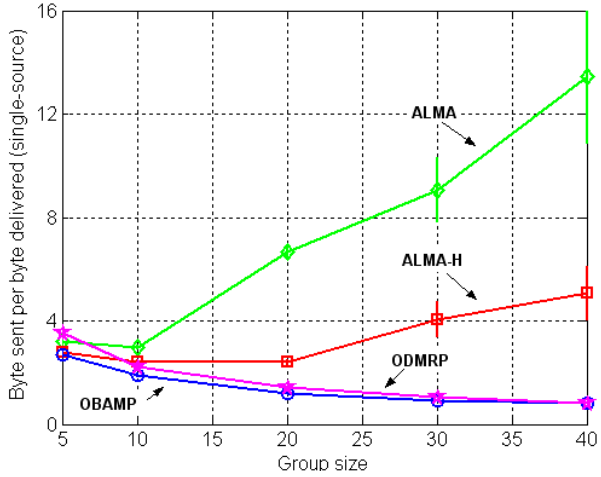


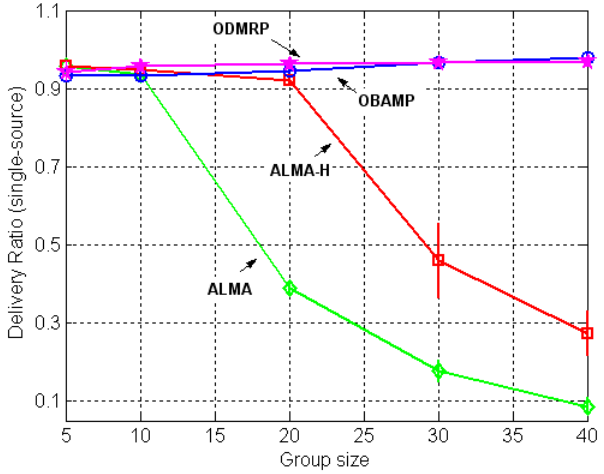



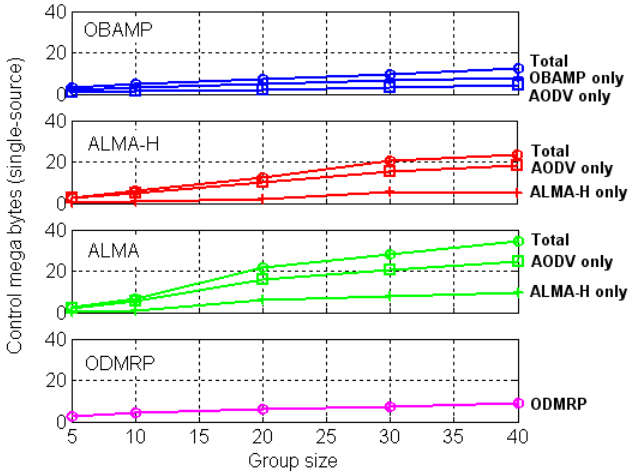



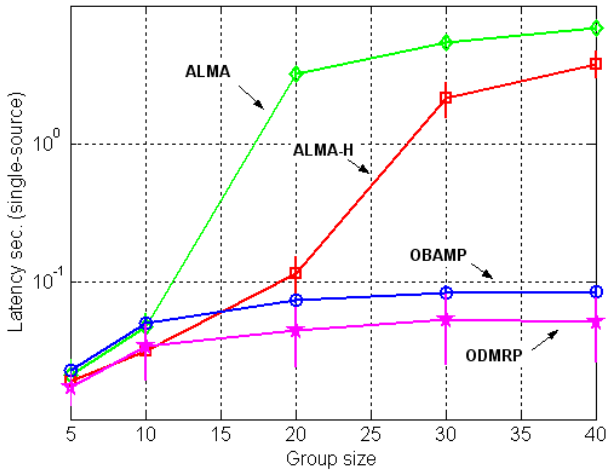



# Single source

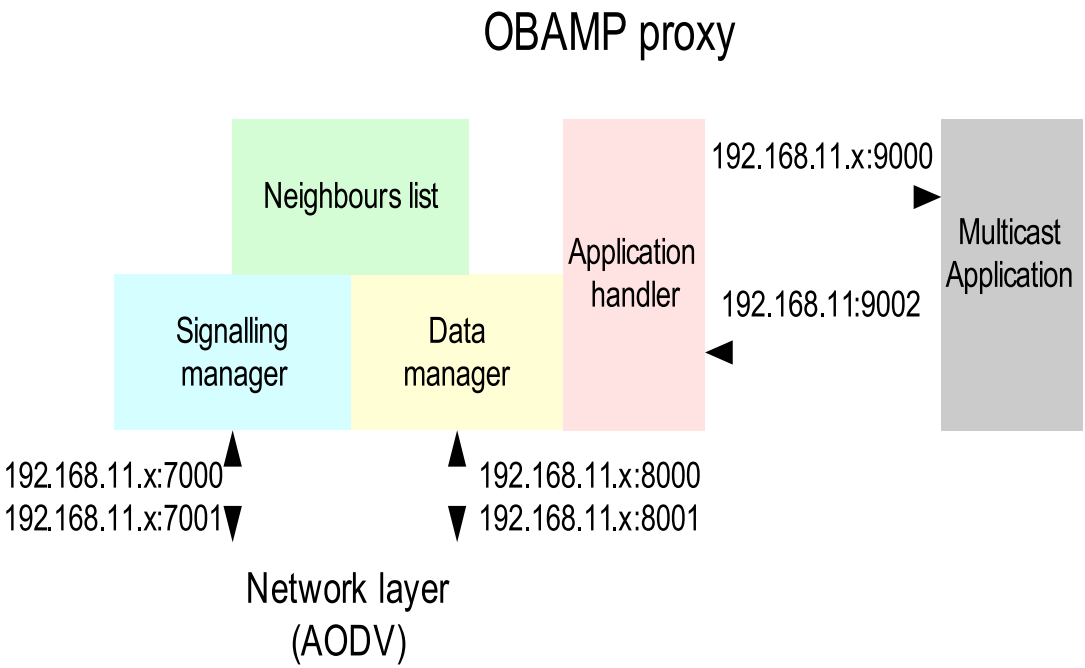


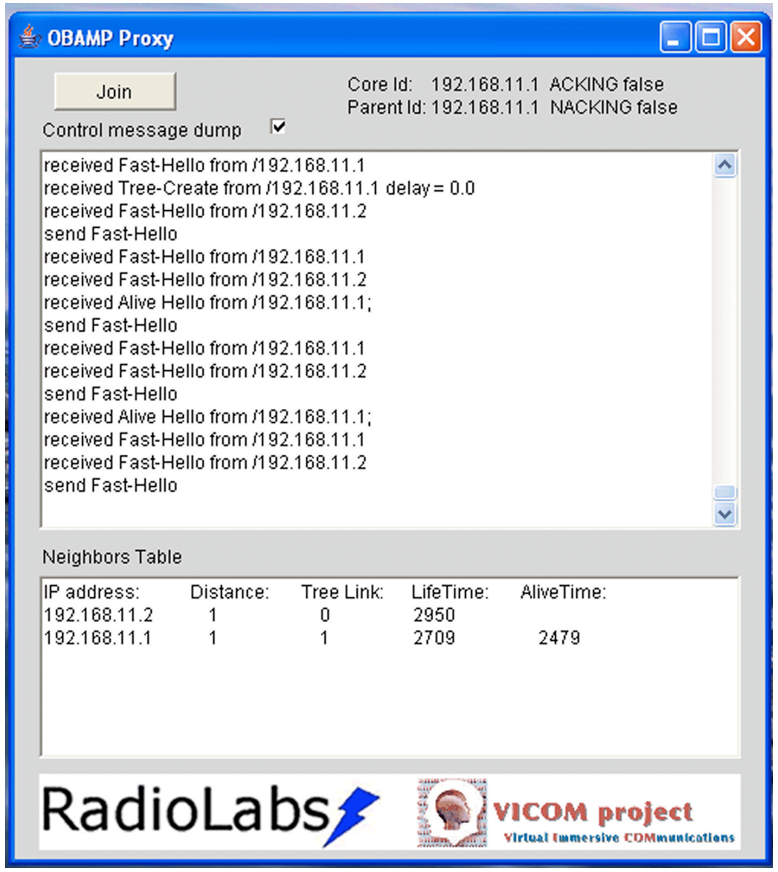



# Implementation



# Testbed - AODV

- 4 nodes
- 3 members (1,3,4)
- 802.11 adhoc mode
- AODV
- Node movement emulated through dynamic MAC filtering

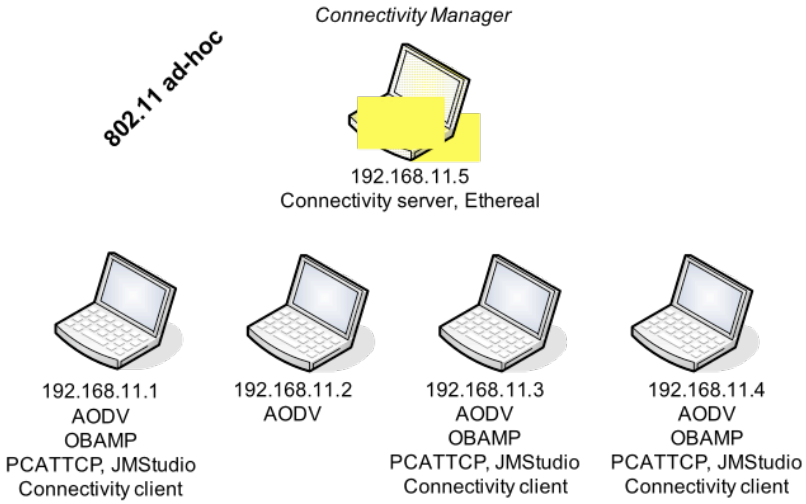
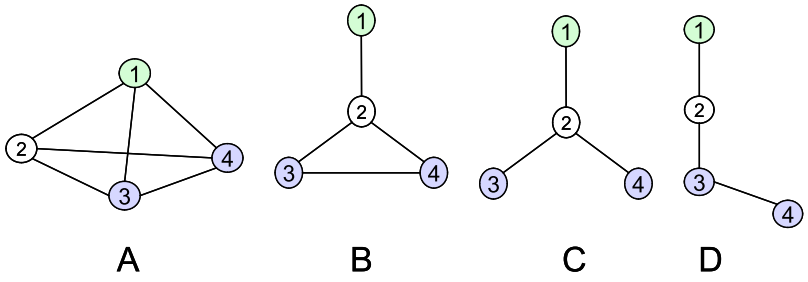
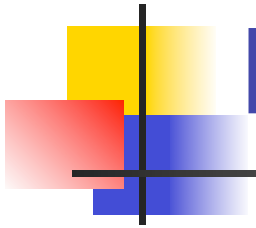



TABLE I
AVERAGE NETWORK BITRATE (KBPS) FOR OBAMP, FLOODING AND UNICAST VERSUS DIFFERENT CONNECTIVITY PATTERNS (A,B,C,D) IN CASE OF AODV ROUTING AND 32 KBPS CBR SOURCE ON NODE 1

| Protocol | A | B | C | D |
|---|---|---|---|---|
| OBAMP | 42 | 119 | 157 | 118 |
| Flooding | 149 | 149 | 149 | 149 |
| Unicast | 77 | 152 | 152 | 189 |



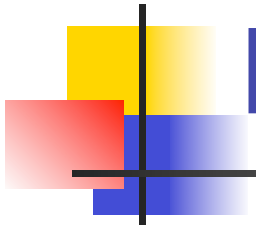


# Porting to OLSR

- OLSR lacks flooding!
- Ninux.org claimed a OBAMP version that worked on OLSR -> **OBAMPxP**
- Node discovery obtained from file
  - actually with the recent BMF plugin this can be fixed!
- Routes gatherered directly from routing table (cross layer OBAMPxP)

# Testbed - Ninux.org (OLSR)

- The testbed on a real mesh network!
- It worked!
- Still be we need to optimize the code
- Some interesting problems emerged!

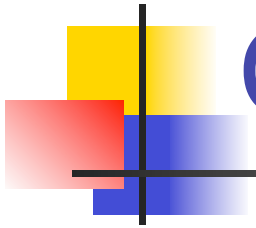


# Real scenario problems

- Broadcast packets are slow because of basic rate
  - High rate content suffers bandwidth bottleneck
- Code performace, code needs more speed
  - Only audio streaming is actually possible

# Source Code GPL

- Mantained now at Ninux.org
  - http://test.ninux.org/ (SVN Repository)
  - http://wiki.ninux.org (Documentation)
- We need help in porting to Mac OS X and BSD
- We need help in optimize
- If you think this is a cool project you are welcome to help!



# Conclusion

- Thanks for your attention!
- Questions?

- Try it and give us feedback! :)