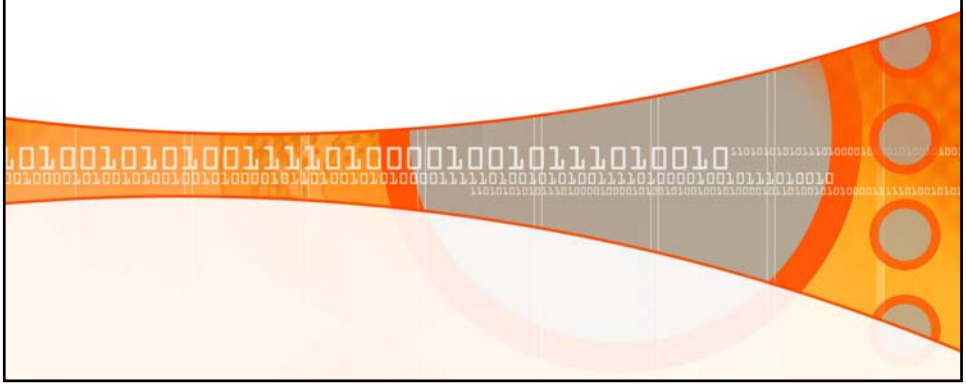
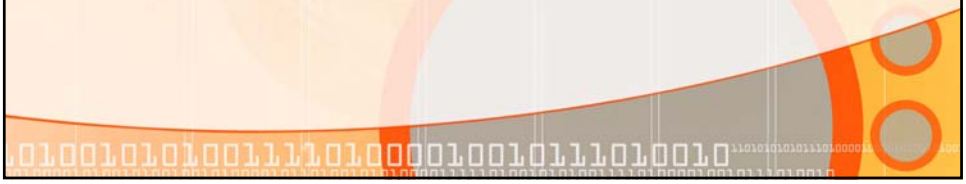
Reliable IP Multicast

Adam Ryan
Michael Henderson



Basic Multicast

- One-to-many connections
- Many-to-many connections
- Possible uses:
  - Streaming video and audio
  - Software distribution
  - Online gaming



Unicast vs. Multicast

- Unicast
  - TCP packets
  - Reliable transfer of data
  - Inefficient for large scale communication
- Multicast
  - UDP packets
  - Easy transfer of data between server and client
  - Requires too much error correction and monitoring to be effective in today's Internet

TCP vs UDP

- TCP (Transmission Control Protocol)
  - Reliable transfer
  - Dependent travel
- UDP (User Datagram Protocol)
  - Unreliable transfer
  - Independent travel

Multicast over WLAN

- Makes good use of bandwidth for each receiver that joins a multicast group
- Must use some kind of algorithm to retransmit data which a client might not properly receive.

IP Multicast

- Uses a Class D IP address (224.0.0.0 – 234.255.255.255)
- Allows servers/clients to join and drop the connection at any time
- Data is transferred to the IP address by the server and retrieved by the clients

Background

- Original Goal: Add multicast ability to Classroom Presenter and test it over a wireless network
- Classroom Presenter 2.0 already has an IP multicast feature
- Easier to measure reliability with our own program

Reliable IP Multicast

- Goal: To test a simple IP multicast setup within the test bed over a wireless network
- Things we measured:
  - Reliability
  - Efficiency
  - Effectiveness of error correction/detection

The Test Bed

- 8 computers
- 1 sender and 7 receivers
- Windows XP Professional
- Microsoft .NET Framework 2.0

MCast

- Developed in C#
- Consists of 2 parts
  - MCSend – server end
  - MCReceive – client end
- Features:
  - Sends strings of data over a LAN or WLAN connection.
  - Pseudo-NACK algorithm
  - Measure time it takes for sender to send a message
  - Implements sockets
  - Records data.

MCSend

- Important Functions:
  - Initializes multicast connection with a multicast address and multicast port
  - User input
  - Sends messages to the multicast group
  - Receives NACK messages via thread initialized within the program via port 65535

MCReceive

- Functions:
  - Receives messages from the multicast connection
  - Sends a NACK message via a Unicast connection between the sender and itself if it does not properly receive a message
  - Records results

NACK

- Negative acknowledgement
- Triggered when a client fails to receive a packet instead of when a client successfully receives a packet
- More efficient than ACK
- Other implementations include TRACK and SuppACK

NACK: MCast

- pCount
  - Counter within MCSend
  - Separate counter within MCReceive
  - Starts at 0 and increments every time a message is sent over the connection
- NACK implementation
  - pCount is appended to each message and sent to a client
  - When a client first receives a message, its pCount is initialized to the same number appended to the message received
  - pCount then increments independently from the MCSend pCount
  - Once a client fails to receive a message over the network, the program becomes out-of-synch with the multicast connection and starts to NACK

Testing

- One test, Two networks
- Goal: To have all clients send NACK messages to the server at the same time
- Measure: Number of “NACKing” clients at pCount (time)
- Tested of LAN and WLAN networks

Assumptions

- No outside interference
- LAN and WLAN connections weren’t mixed

The Test

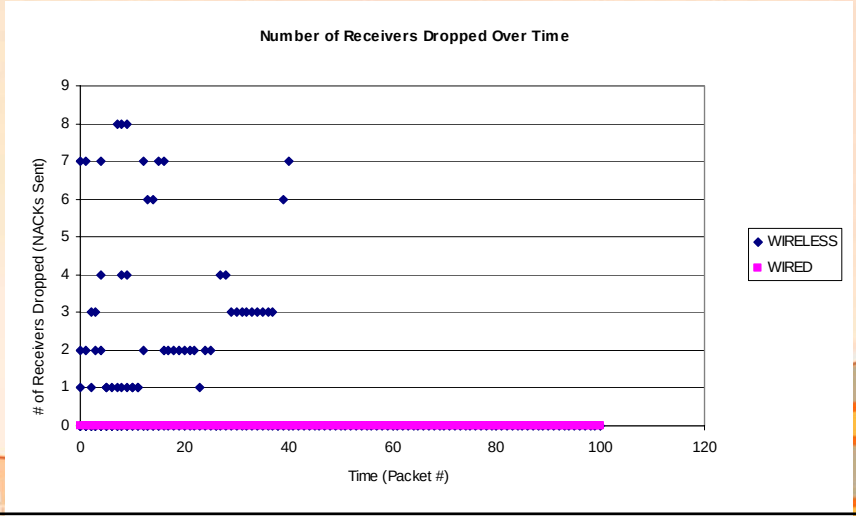
- Sent a few test messages to make sure everything is working properly
- Sent different lengths of messages
- Try overloading the connection by sending a constant stream of messages to the multicast address

LAN Results

- Could not break it
- No client sent a NACK message
- Longest run: 584 messages

WLAN Results

- Broke very quickly
- Unpredictable results



Conclusion

- While no new information about IP multicasting over a WLAN connection was found in the duration of this project, we have reiterated that this method is still unreliable due to its use of UDP packets and error correction methods which are either inappropriate or not developed enough. However, IP multicast still remains to be the standard today.

Problems with MCast

- No way of retransmitting the lost packets
- Inefficient yet effective algorithms
- The only thing that one is able to read from the logs is the number of NACKing clients at the time
- Logs not formatted
- Bug where MCast prints out the wrong Unicast IP address and port

Project Problems

- Confusion
- Not enough resources for a large scale test (N > 30)
- Not enough participation for a large scale test due to lack of resources.
- Alternative methods were researched but not used.

Looking Ahead

- Some things we could do in the future:
  - Implement Classroom Presenter with different types of multicast
  - Fix MCast and develop it further as a monitoring tool
  - Launch MCast as an open source project
  - Play around with WLAN settings for optimal results

Project References

- Association of Computing Machines (ACM)
- IEEE
- Internet Engineering Task Force (IETF)
- Wikipedia
- Dr. Haibin Lu
- Dr. Wenjun Zeng

Acknowledgements

- Dr. Lu & Dr. Zeng
- The other REU participants
- Mizzou
- Payroll office