# A Comparative Study of Application Layer Multicast Protocols

As the title suggests, "A Comparative Study of Application Layer Multicast Protocols" by Suman Banerjee and Bobby Bhattacharjee focuses primarily on the comparison between existing application-layer multicast protocols. The article provided a brief introduction on the need of application-layer multicast protocols over existing IP multicast protocols, and further reviewed application-layer multicast protocols from three distinct categories: mesh-first, tree-first and implicit approaches.

The article provided little and very generic reasoning on why application-layer multicast is a better alternative to IP Multicast, which a long existing and established network-layer mechanism. The inclusion of such information could greatly strengthen its argument that application layer multicast is the better choice for content distribution to more than one host. After all, in contrast to unicast where a separate copy must be sent for each receiver, multicast lets network routers handle the distribution process and sends only a single copy. With multiple data streams now being replaced by a single stream, it is theoretically the most efficient method for group file distribution. IP multicast can effectively utilized network bandwidth and therefore ideal for Internet video and audio broadcast, video conferencing, online gaming, file distribution, and large system maintenance such as dispatching update over remote hosts.

Despite its benefits and potential usage, IP multicast has suffered limited deployment largely due to the following reasons:

1. Reliability: Most IP multicast applications are UDP based, therefore, packet drops happen and are to be expected. While an occasional lost of packet is tolerated for video and audio streaming, file distribution requires reliable delivery.
2. No Congestion Control: largely due to the nature of UDP, IP Multicast can cause overall network degradation as multicast applications start to grow.
3. Overhead and complexity at the routers: routers need to store per group state as well as an entry for each multicast group address in the routing and forwarding table. Such addresses are not easily aggregatable.

Application-layer multicast was proposed as an alternate technique for multicasting due to these reasons. It provides the exact same functionality as IP Multicast, only data packets are now replicated at end-hosts rather than at the network routers. The article classified application layer multicast protocols into three different categories based on the way they organize the group members: mesh-first, tree-first, and implicit approaches.

In mesh-first approach, each group member exchanges periodic messages that include group membership and routing information. This approach is highlighted by the source-specific tree that is rooted at each of the participated group member. The mesh-first approach protocol discussed in this article was:

1. Narada –Using a designated host called the Rendezvous Point, new member can join by obtaining membership list and sends out requests. Membership and routing information are

then propagated to all members where each member can then compute unicast paths between all pairs on the mesh.

The article mentioned the shortcoming of the Narada protocol as its effectiveness can only be seen in a small multicast group because of the large overhead resulted from storing membership state information at each member. However, the article did not link such shortcoming to the potential consequences where the protocol could drastically limit the range of suitable application including popular video broadcasting, online games, and file distribution.

As mentioned in the paper, because of the nature of mesh-first approach, the protocol produces near optimal routes and has possible the best end-to-end delay out of all application layer multicast protocols. It is achieved by requiring each group member to individually compute unicast paths to the rest of the group. This requirement pushes additional processing requirements and overhead on the end host. One possible solution could be to have each member calculates the cost to its immediate neighbor then exchange such routing information with the rest of the group to reduce overhead.

In tree-first approach, a shared data delivery tree is distributedly constructed first. Each member discovers a few other members that are not its neighbors on the overlay tree and maintains additional links to them. Tree-first approaches discussed in this article include:

1. <u>Yoid</u> – Yoid, like all other tree-first protocol, has a shared tree and is in control of its structure. New member queries the RP for membership list to join, and is responsible for finding its appropriate parent on the tree. A member becomes the root of a new tree when it's unable to find an appropriate parent, and now it becomes the RP's responsible for merging different trees.

   In addition to overcoming the fundamental issue of IP multicast by providing a reliable and congestion controlled approach using TCP, Yoid is able to scale to large environments unlike the previously mentioned Narada protocol.

   Despite its strength, Yoid suffers from a few shortcomings and some of them are not disclosed in this article. The shared tree could be far from optimal in a large scale environment and its creation would result in large overhead. An increase of end-to-end delay is expected with group members taking suboptimal routes.

2. <u>HMTP</u> – HMTP shares many similarities with the Yoid protocol. However, major differences include new members join from the root rather than with any random member, members maintain information about all members on its path to the root, constant optimization, and loop detections.

   HMTP suffers from a similar shortcoming as the Yoid protocol: suboptimal tree. When new member is attempting to join the tree, the protocol is looking for potential parent using first fit model rather than best fit. In addition, the loop detection feature would create a large overhead especially in large environment and might not be complete. A loop in the shared tree could result in data duplication and low network utilization.

Implicit approach, unlike tree-first or mesh-first approaches, creates a control topology where tree and mesh are simultaneously defined by the protocol. No additional member interaction is needed. Implicit approaches discussed in this article include:

1. NICE – The protocol arranges a set of members into a hierarchy that defines both the control and data overlay topology. Such hierarchy structure is essential in achieving scalability and limiting effect of member departures or failures. New member joins the hierarchy that is closest to itself with respect to the distance metric, which allows the production of better trees with more optimal paths.

   One of the greatest strength of the NICE protocol is its hierarchical structure which enables its great scalability. However, it comes with the price of efficiency. Both the creation and maintenance of the hierarchy as well as the storing of members' state generate large overhead. In addition, the complex joining procedure further reduces the efficiency and performance of the protocol.

2. CAN-Multicast – CAN-Multicast is an approach based on the Content-Addressable-Network whose constituent members form a virtual d-dimensional Cartesian coordinate space. Data packets are forwarded from the source to its control topology neighbors.

   Scalability is one of the strengths of CAN-Multicast. There seems to be no physical limitation on the group size because the service model is not restricted to a single source. The protocol avoids loops in its structure with a well defined forwarding rule. However, some drawbacks include high stretch overlay paths because neighbors on the CAN may be far apart. In addition, each member is required to maintain a packet cache to identify any duplicate packets. Such requirement would post a large overhead especially when large amount of data is being distributed.

While application-layer multicast enjoys the benefits of requiring no infrastructure support, easy deployment, and reliability and congestion control from TCP, it has the drawback of inefficiency when compares to network layer multicast. The fact that application layer lacks knowledge about underlying network topology will result in performance penalty for all application layer multicast protocols. This would generate lower efficiency and longer end-to-end latency as applications must now measure the network topology themselves. In addition, the application layer multicast protocols presented in this paper in general has large control overhead for maintaining member state. With the unsuccessful deployment of IP multicast, it might be worth a shot to try application layer multicast with its upsides and potentials. However, their issues will remain to be even greater challenges with the growth of application layer multicast deployments.

Overall, the paper provide no novel idea or evaluation but rather just simple generic overview of different application layer multicast protocols. It did not mention how application layer multicast protocols were able to overcome the difficulties faced by IP multicast. In addition, when evaluating each protocol, the paper failed to consider the quality of multicast tree which would indicate the bandwidth usage and end-to-end delay.

The evaluation results in this paper were gathered based on theoretical values and speculations rather than experimental data. An extended idea to this paper would be to perform actual deployments of each application layer multicast protocol and conduct an in-depth comparison between each or even with existing improved IP multicast protocols.