Peer-2-Peer and GRIDS
Reading Assignment 02: SplitStream

Motivation

In traditional tree-based multicast systems a large number of clients are serviced by a relatively small number of interior forwarding nodes. Although this can work well in specialized setups this is inherently unfair, vulnerable to failure and poses some scalability problems.

Contribution

The authors propose SplitStream, an application level, tree-based multicast system which fairly distributes forwarding/pushing load to all participating nodes, thereby utilizing total available bandwidth more efficiently. Additionally, SplitStream's design can be robust with regard to failures. An important part of their contribution is that these features can be achieved with relatively small overhead.

Solution

As stated in the motivation, a small number of interior nodes have to service a large number of client leaf nodes. This setup places undue load on these interior nodes and doesn't utilize the upload capacity of the leaf nodes. Increasing the depth of the tree to lower the load on each interior node is not the solution since that method increases latency and susceptability to failures. SplitStream approach to tackling this is quite simple. The content is broken up into multiple stripes and for each stripe there is a separate multicast tree through which the stripe is pushed, forming a forest of trees that distributes the forwarding load. SplitStream aims to efficiently construct this forest of trees, in a decentralized and self-organizing manner, where the vast majority of nodes are interior nodes in only one tree, while also respecting bandwith constraints of nodes by having specific adoption rules.

Tolerance towards failures is achieved by having these multiple trees, with each node most propbably being an interior node in only one, and by using a suitable data encoding. A failure of one node means that one stripe will fail and the data encoding should ensure that this only means degradation in quality, not a complete loss of usability.

The author state that SplitStream provides a generic infrastructure for high-bandwidth content distribution. SplitStream itself is also generic in that it can be implemented with various tree-based multicast schemes, and applied in situations where two conditions are met. The first condition is that total demand doesn't exceed supply capacity. The second condition is that nodes unable or unwilling to forward stripes don't hold up the distribution.

The authors use Pastry and Scribe as an example implementation for illustration. We will refer to that setup as well.

To create these so called interior-node-disjoint trees SplitStream takes advantage of Pastry's prefix routing by using Scribe tree/group/stripe Ids that differ on the most significant digit. Since the group is "based" on the most significant digit, those which differ will become leaves/clients of this particular stripe group, and not interior nodes.

To join this distribution network a node seems to have two options, either that it knows an exising node(s) or that it uses Scribe's anycast feature to get in touch with a node in a special group called Spare Capacity Group, which contains nodes that can accommodate additional clients. Respecting bandwith constraints is then done by with an "other-tree-aware" variation of Scribe's push-down scheme and the spare capacity group. When nodes join groups they are pushed down in the tree until a spare capacity node is found, with preference given to nodes with matching prefix. If nodes become orphaned in this process they can use the Spare Capacity Group, or ask siblings, to find another supply tree/group for the stripe.

Evaluation
The experimental evaluation seems mostly well thought out and performed. A large number of nodes were used, both in an event simulation situation and in live experiments in PlanetLab, to test factors such as node stress, link stress and delay penalties. The multiple trees of SplitStream are design choice which might appear to be expensive so the behaviour in that regard was evaluated. Content distribution, i.e. the multicasts themselves, was also tested, as well as behaviour under both massive failure and high churn. The performance was also compared to other multicast methods, those of IP multicast and a single-tree Scribe.

The solution of multiple trees seems effective, with distribution of load in the form of very high utilization of links in the topology, while considering nodes' bandwidth constraints, and without large overhead in forest construction or state. Authors mention that overhead of forest construction and distribution of 1MB of data with SplitStream's approach is significantly less than the cost to a centralized server distributing the same data.

Disadvantages of Solution
Authors mention problems when the bottleneck isn't at sender or receiver, but somewhere in between, and how that affects bandwidth considerations and forest construction. They plan to extend SplitStream to tackle this issue.

Disadvantages of the Evalution
In testing of high churn authors sent a data packet every 10 seconds. This interval seems very long if we think of content distribution in terms of live-video streaming or similiar. One wonders what the effects of high churn are in higher demand situations.

Further Improvements and future directions
Authors mention that dealing with free-loaders in general, and in SplitStream in particular, is an interesting area of future work. Authors also mention problems when the bottleneck isn't at sender or receiver, but somewhere in between. One would however think that the choice of implementing platform could help there, for example Pastry's network proximity awareness.