BCBS: An Efficient Load Balancing Strategy for Cooperative Overlay Live-Streaming

Thorsten Strufe and Günter Schäfer
Department of Telematik and Computer Networks
Technische Universität Ilmenau
Langewiesener Straße 37
D 98693 Ilmenau

Arthur Chang
Department of Information Management
National Taiwan University of Science and Technology
43 Jilong Rd, Da-An District
TW Taipei City 106

***Abstract*— In this paper, we present Bandwidth Class Based Streaming (BCBS), an application layer multicast for multimedia services. BCBS focuses on multi source live streaming, and following a locality model based on round trip times, it creates network efficient streaming meshes. The load balancing selects multiple nodes as streaming sources and is organised subscription based instead of request based. We describe a simulation study of the load balancing and tree construction procedures. The results show that BCBS creates network efficient overlays with respect to stretch and link stress.**

I. Introduction

Streaming multimedia poses a high bandwidth load on the service provider. In the commonly used client/server-like setup this can lead to a very high cost for the provider or even to service failures due to a bandwidth squeeze at the server link. Different load balancing schemes and communication models have been implemented to solve this problem. Network multicast[1], server farms and content delivery networks (CDN)[2] however can commonly not be used, especially by small service providers. Network multicast on the one hand suffers from a low acceptance and deployment through the ISPs while CDN and server farms on the other hand pose high cost on the service provider and maintainer. With overlay multicast a different load balancing scheme has been introduced, which is self-provisioning in the sense, that every participant for using the service serves other participants in turn.

Two different scenarios of multimedia streaming are video on demand and live streaming. In a video on demand scenario the content is preproduced and provided by an arbitrary number of source nodes. Later parts of the stream can be downloaded in advance and buffered locally, if bandwidth is available. In live streaming on the other hand, the content is produced as the users are consuming it. It has strong realtime characteristics and all participants have to forward the content as soon as they receive it. Flash crowds, a high number of users joining a system during a short period of time, can especially be observed in live streaming scenarios, when an event of major interest occurs.

A simple overlay live streaming model consists of three types of nodes: a source, one single end node which is providing the content, requesting nodes, which create the streaming overlay forwarding the content, and backbone nodes (plain IP routers). As a large fraction of end hosts in the Internet is connected through asymmetric digital subscriber line links (DSL), the overall downlink capacity is a lot higher than the overall uplink capacity. In opposition to highly available content servers, overlay nodes are rather unreliable. Due to users joining and leaving the service and to parallel traffic, the participation time and available bandwidth of the overlay nodes are highly dynamic.

In order to still create a cooperative live streaming overlay, some requirements have to be met.

The most important requirements for a cooperative streaming system are stability and network efficiency. One main problem of overlay streaming, and other decentralised distributed systems as well, is the decoupling of the overlay topology and the underlying infrastructure network. This leads to inefficient overlays with long routes between nodes which are more susceptible to QoS problems. In addition they generate high traffic load of redundant packets on the backbone. Hence a topology aware overlay should be constructed in order to avoid redundant traffic and to make use of short links, which are less susceptible to packet loss, jitter and failure. A simple means of creating overlays and being able to quickly switch to a different source is to serve every sink from only one source node. However, to be able to create stable overlays with short links, it is important to distribute the load on multiple sources and to make use of low uplink capacities. For reasons of fairness and stability the load should be distributed equally over the participating nodes as well. This objective additionally leads to a lower risk of congestion on single links due to a high volume of parallel traffic.

Overlay streaming protocols, which are used to construct the streaming overlay, follow the three steps of acquiring an initial entry point, locating potential source nodes, and constructing the overlay. The protocol overhead should be low to avoid unnecessary traffic and processing. In order to be able to handle large groups of participating nodes, the overlay has to be constructed through a distributed algorithm, based on information about parts of the overlay only. While a centralised management has many advantages when it comes to optimising the resource usage, it introduces single points of failure and potential performance bottlenecks. Distributed

algorithms which need an overview of the whole system[3] do not scale to large groups due to their message- and space complexity.

With Bandwidth Class Based Streaming (BCBS) we present an algorithm, which focuses on multi source live streaming and creates efficient streaming meshes. BCBS follows a locality model, which is similar to Vivaldi [4] and always selects the closest sources to create a topology aware overlay, leading to better network efficiency and QoS.

BCBS neither selects single sources nor uses layered streaming. Multiple nodes are selected as sources and load balancing schemes over sequences of RTP-Packets are calculated instead. The load balancing is done subscription based, describing filters for every source, instead of requesting chunks of the multimedia stream. As a consequence it has a marginal protocol overhead compared to the video data and can be used for live streaming as well as video on demand applications.

The rest of the paper is organised as follows: section two contains related work to load balancing in overlay streaming systems, in section three we describe our approach called *Bandwidth Class Based Streaming (BCBS)*, section four describes a simulation study of our scheme, and in section five we draw some conclusions and describe future work.

II. BCBS

Bandwidth Class Based Streaming is a multi source load balancing algorithm for live streaming. In order to avoid redundant traffic and to keep the traffic to local networks instead of creating a high cross domain load, the main design goal was network efficiency. Additionally the algorithm is required to scale to large groups of nodes. It should furthermore lead to a fair distribution of the load over the participating nodes. As the end hosts are assumed to be fairly unreliable and a node failure should not lead to a complete service breakdown at the following nodes, the algorithm should create meshes with every sink being served by multiple sources.

BCBS is divided into a set of functions on a requesting sink node(see also the algorithm description in Figure 1) and a set of functions on the source nodes. After entering the overlay through detecting and connecting to a first node, which already participates in the system, a new sink locates sources and subscribes to the service. The sources on the other hand implement an admission control and from the information in the subscription message calculate filters to select the packets which they will forward to the sink.

To locate potential source nodes or different multimedia services, BCBS relies on a locality aware P2P search algorithm, which we have already described in[5]. This location service creates clustered locality aware topologies which can be used for signalling. Every cluster includes a super node which implements a registry service for the local nodes and which are interconnected through a peer to peer network. The implemented locality model is similar to Vivaldi [4] and PIC [6] and round trip times are used as a locality measure.

Following the approach of [4], every node keeps a seven dimensional vector of its own position, which is updated through regularly keeping track of the round trip times to the neighbour nodes.

When a new node joins the system it searches for seven random super nodes and performs a locality estimation through measuring the round trip times to the known hosts. After storing the results as initial values of the locality vector, the super node with the shortest distance is located and chosen as the local registry service. In case that a cluster grows above a limit in size a MINCUT algorithm is run by the super node with the distances between nodes used as edge weights. The node which is closest to the centre of the new cluster (according to the locality measure) is selected to be the super node of the new cluster and the local nodes register themselves with their new registry service. As the super nodes are informed on resource updates by the local nodes, they can reply to local resource requests very quickly.

BCBS()
1 $SourceSet \leftarrow locate(localVicinity)$
2 **for** $node_i \in SourceSet$
3 **do** $UpdateBW(node_i)$
4
5 $sort(SourceSet, distance)$
6 $ClassTable \leftarrow calcTable(SourceSet, R_0 * 1.2)$
7 **for** $node_i \in ClassTable$
8 **do** $subscribe(node_i, ClassTable)$

Fig. 1. Algorithm Bandwidth Class Based Streaming

After locating a set of nodes which are registered as source nodes of the stream through the location service, a subset of nodes is selected as the original source set. In this step, only nodes of a lower generation are selected, in order to avoid loops in the mesh. The generation is simply determined from the stream time of the last received packet to keep the management overhead low. The sink node requests updates on the generation of the nodes in a local vicinity, in case that they had to reschedule in the meantime and uses the information obtained from this as an estimation for their available bandwidth and distance. Nodes with the lowest distance, measured through their round trip times, are selected, until the sum of the available bandwidth is higher than the bitrate of the stream. To avoid service failures even in the case that one node alone is able to support the whole bandwidth, a set of at least three local nodes is selected, if available.

Before subscribing to the selected sources, a table of exponential virtual bandwidth classes is created (see also Figure 2). The highest class is determined by $\lfloor ld(bw_{max}) \rfloor$.

The available bandwidth of every source is split into these classes and the source node annotated accordingly (see also Table I), with every node of a certain class being able to send twice as many packets as a node in the class below. When the sum of the bandwidth has reached the bitrate of the requested

```
CALCTABLE(SourceSet, R)
 1  class ← 0; bwclasses[]; i ← 0;
 2  helpSet ← sort(SourceSet, bw)
 3  while pow(2, i) < helpSet[0].bw
 4  do
 5      bwclasses[i] = pow(2, i)
 6      i++
 7
 8  reverseSort(bwclasses[])
 9  for class_i ∈ bwclasses[]
10  do for node_j ∈ SourceSet
11      do if node_j.bw ≥ class_i
12            then
13                 class_i.push(node_j)
14                 node_j.bw− = class_i
15
```

Fig. 2. Function to create the Virtual Bandwidth Class Table

stream and at least three sources are selected, all nodes with a lower (absolute or remaining virtual) bandwidth are ignored and the remaining classes are stripped from the table. The resulting table of bandwidth classes and sources is sent to all selected source nodes together with the subscription message.

Class	SourceNode
Class0 = 256kbps	Node1
Class1 = 128kbps	Node2, Node3
Class2 = 64kbps	<noNode>
Class3 = 32kbps	Node1

TABLE I

TABLE OF VIRTUAL BANDWIDTH CLASSES

Upon reception of a subscription message, every source node checks, if it is able to support the requested bandwidth to grant or deny the access of the requesting node. From the table the source calculates a filter containing the local schedule of packets which have to be sent to the sink. The calculation of the filters follows two steps.

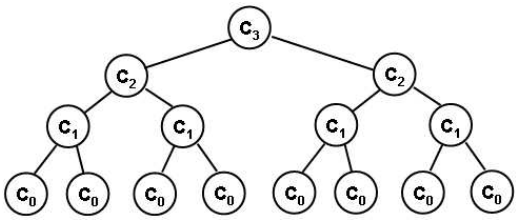

Fig. 3. Bandwidth-Class Binary Tree

At first, every node creates a reverse binary tree of the bandwidth classes(See: Fig 3), annotating the nodes of every class to each node or leaf of the tree(See: Fig 4).

To calculate the well described schedule of every node, it only has to traverse the tree and build the sequence of nodes of

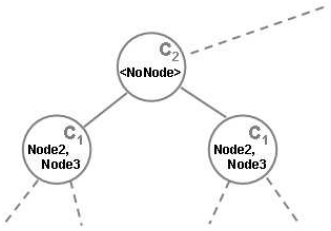

Fig. 4. Part of the binary tree

the whole tree. The filters are simply derived from recording every instance of the local node in the sequence and the overall length of the sequence as the basis for a modulo calculation over the sequence numbers of the RTP-packets in the video stream(See: Fig 5).

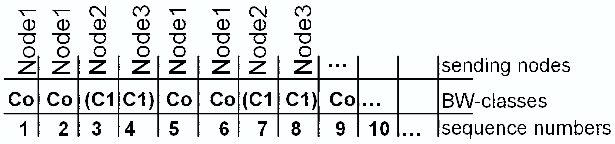

Fig. 5. Schedule derived from a postfix traversal of the binary tree

III. PERFORMANCE STUDY

To examine if BCBS meets the requirements and creates network efficient topologies, we conducted a simulation study with changing group sizes, using OMNeT++[7]. In order to use an underlying topology which in its characteristics resembles the Internet, the networks were generated using BRITE[8]. The topology generation followed the model of [9] and a backbone of 500 nodes was created. The backbone network had a diameter of 7 hops and a characteristical path length of 3.455. A stream with a bitrate of 500 kbit/s, an uplink bandwidth of 100mbit/s for the original source and of 100mbit/s, 1mbit/s and 500kbit/s in three different simulation setups for all end hosts were assumed. The group sizes of participating nodes were ranging from 50 to 1500 end hosts. To follow a flash crowd characteristic all end hosts requested the streaming service at a random point during the first 10 seconds of the simulation and the simulation was conducted for another 1000 seconds to make sure that all load balancing sequences were fully traversed and all relevant data gathered.

In order to capture steady state behaviour, the measures were taken after removing the transient phase. To show the efficiency of BCBS we chose the metrics link stress and packet stretch.

A. Stress

To judge the efficiency of the created overlay topologies and to investigate how much data is transferred redundantly, the link stress[10] was examined.

Link stress is defined as the amount of identical packets traversing each link.

To gather the link stress, the mean number of identical packets was logged for every link during the simulations and an average calculated afterwards.

High stress values in the backbone on the one hand are undesirable, as they show an inefficient use of the resources. High stress on end hosts' links on the other hand are unacceptable, as they typically only have a low capacity. Overlays with an inefficient topology lead to high link stress in the backbone, as many packets are transferred forth and back over the links of the network infrastructure.

When examining the stress, two different values are of interest, the worst case, or maximum link stress and the average link stress.

The maximum stress at the end hosts rises if the load balancing scheme creates unfair schedules, as many end hosts experience very low links stress only, while the whole load is left to be served by only a few other end hosts. The average link stress can be taken as an indication of how much redundant traffic is transferred in the network and thus how well the overlay resembles the topology of the infrastructure.

Each joining node receives the stream once in full and causes a link stress of one on the incoming links. If every node forwards the whole stream to one neighbour node, the stress on the links to the end hosts will average to two. Considering a completely switched network resembling a tree, where every leaf is receiving the stream (and the amount of leafs is significantly higher than the amount of inner nodes), the overall average link stress (in this case the absolute stress at every link) is converging to two for a protocol constructing good overlays. When meshes are introduced and short-cut links added to the infrastructure the stress will converge to two even quicker, as possible multi hop paths (with a stress of one on every link in between) will be avoided. However, in the realistic case a network has a high amount of backbone links, compared to the number of members in the multicast group. This fact leads us to the belief that an average stress of 2 should not be exceeded for a good overlay construction protocol.

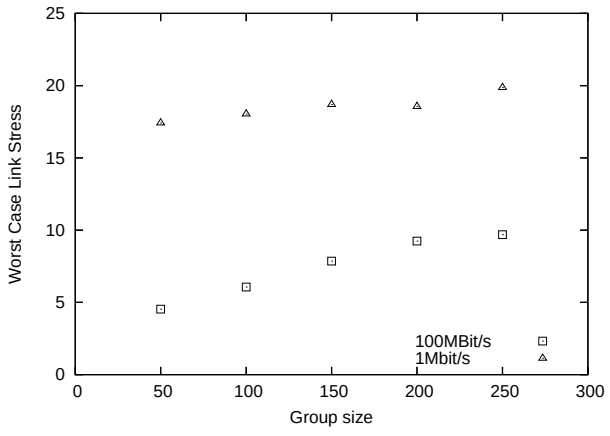

Fig. 6. Worst Case Link Stress for small groups of nodes with 1Mbit/s and 100Mbit/s uplink capacity

First, we investigated the overall maximum link stress (See Fig: 6) for the different setups. In all simulation runs of BCBS the worst case link stress was observed on links in the backbone. For the simulations with lower initial available bandwidth at the overlay nodes, the average stress at the uplink of the original source rose. However, the initial available bandwidth of the source was never completely used. Even though BCBS is organised without a central management and without a total view of the system, the examined maximum link stress for all group sizes is low compared to other systems and even lower than for example in the simulations of Narada.

For naive unicast the maximum stress was measured at the uplink of the original source as expected. It served all sinks and therefore had to deliver the highest rate of identical packets.

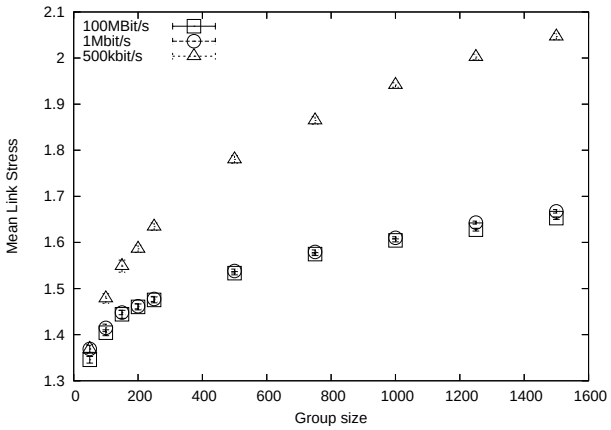

Fig. 7. Average Link Stress with 95% confidence intervals

To examine the locality model and matching between the overlay and the underlying infrastructure network, the average link stress was calculated for all simulations as well (See Fig.: 7). The rise of the mean stress in the simulation setups with high capacities at the end hosts was less than linear and hence scaling very well. Only for the simulation setup of end hosts with low uplink capacities, in which every node was able to forward the stream only once, the mean stress actually exceeded two. However, this result is expected, as the nodes joined randomly all over the network and BCBS was only conducted at the time of joining. This behaviour leads to high distance links, as usually a node with a long route has subscribed to some sources before close-by nodes join the system. An optimisation of the overlay could be reached by simply doing cyclic runs of BCBS in order to find closer nodes in the next iteration.

B. Stretch

To examine the overall path a payload packet has to travel before it is received at the sink nodes, we analysed the stretch of the overlay topologies.

Stretch is defined as the overall amount of hops after traversing the overlay topology divided by the number of hops of a unicast link between the original source and the sink.

High stretch is a sign for an inefficient topology and for overlays with high trees or meshes. As the packets travel long routes before they are received at the sinks, it indicates high delays as well. However, especially in networks with small world characteristics[11] the optimisation of the topology in respect to low stress and low stretch are competing goals. In most cases the route to the original source will be significantly shorter than an overlay route with only two or three overlay hops on the way. Even if a packet is forwarded between overlay nodes on short routes, the overall path can still add up to high values.

In addition to being a sign of inefficient topologies, high stretch values can be an indicator for instable overlays as well. In unbalanced topologies, which include parts with very long overlay routes and where a large amount of packets is forwarded along a single chain of nodes as a consequence, a leaving or failing node can cause a significant loss of packets at the succeeding nodes in this chain.

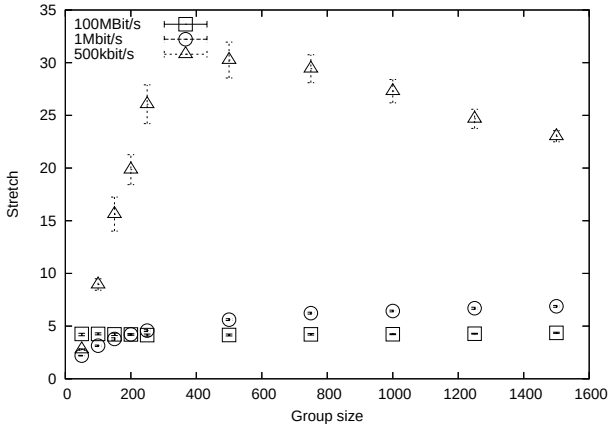

Fig. 8. Average Packet Stretch with 95% confidence intervals

The simulation results of the first setup, with all end hosts being connected through links of 100 Mbit/s capacity, showed an almost constant stretch of around 4.2 (See Fig:8). The high stretch value for a small group size is caused by BCBS trying to distribute the load to nodes with the highest available bandwidth. As in the beginning the distance to all nodes is comparably high, most nodes of the network are potential sources. As a consequence, the nodes that have only just joined the system have packets with higher stretch but offer a high remaining available capacity and are thus selected as sources. With growing group sizes the new nodes locate sources in a short distance and the stretch only rises by small amounts. The meshes stay balanced and as all areas are provided with sufficient uplink capacity it keeps to small heights.

In the second setup, with nodes of 1Mbit/s uplink capacity, the stretch for small groups is lower, as most nodes subscribe a high amount of packets from the original source, which has a very high capacity compared to the other overlay nodes. As short distance sources are located for growing group sizes, the nodes joining later will subscribe to these, thus again keeping the traffic to local areas.

The simulation of smallband nodes, with an uplink capacity of 500 kbit/s only, showed a sharp increase in stretch for growing group sizes lower than 500 nodes and a slow decrease for bigger groups. This increase is caused by the same fact, which leads to high stress values: as the available bandwidth of local nodes is used by long distant sinks, which could not be served by nodes in their own local vicinity, a joining node again has to subscribe to other long distance sources. The decrease is only caused by the fact, that in the simulation the probability of nodes in the same region of the network joining in a short period of time increases for large groups. Using simulation networks with a higher amount of backbone nodes would shift the maximum to a higher number. Again repeating the runs of BCBS during the streaming should reduce this drawback.

These results show that for networks of cooperating nodes with an available uplink capacity which is only close to, or slightly higher than the bitrate of the stream, BCBS will create overlays which are potentially unstable with respect to node failure and may suffer from high delays. However, as soon as the available uplink ratio between available capacity and bitrate rises to only two, the resulting topologies have a significantly lower stretch, even though the stress remains at a low level.

IV. Related Work

Application layer multicast systems can be distinguished according to a few design goals and characterstics. An important criteria is the streaming scenario. Most of the algorithms are aimed at video-on-demand applications, but only a few can be used for live streaming. Other criteria are the decision to choose one single source or multiple sources for each sink and the topology of the overlay (trees vs. meshes). Some solutions for the purpose of having knowledge about all other participants create a signalling mesh, whereas other algorithms decide on local information only.

PROMISE [12], for example, can create very efficient streaming overlay meshes with multiple sources for each sink, but is designed for video on demand environments.

Optimal Multiplexing [13] and PALS [14] on the one hand are creating meshes in order to keep the management overhead low but forward data in advance and can not be used for live streaming as well.

In difference to these three approaches, BCBS is designed for live streaming scenarios and does not rely on preproduced content.

CoopNet [15], Narada [3] and SplitStream [16] on the other hand are examples for systems which are creating multicast trees that can be used for live streaming as well.

Both CoopNet and SplitStream create a forest of multicast trees. CoopNet introduces a centralised algorithm which is only executed when central content servers experience high loads and then makes use of selected high performance nodes only. The centralised nature of CoopNet requires a very performant and reliable machine for the management and introduces a single point of failure. These two issues do not need to

be addressed in BCBS due to its decentralised organisation. SplitStream builds multiple multicast trees for better stability. Each node is an inner node in one tree only and leaf in all other trees, to restrict the impact of a potential node failure to the overall performance. While BCBS creates network efficient topologies, both CoopNet and SplitStream do not optimise the evolving overlay trees for stress or stretch. CoopNet, in order to be able to handle high amounts of participants, does not optimise the tree and such an optimisation would be counterproductive to the design goals of SplitStream.

Narada creates a topology aware multicast tree and minimises the redundant traffic on the net. However, each sink selects only a single source leading to overlays which can not utilise low uplink capacities and which are very susceptible to failures and attacks. An additional drawback of Narada is the overlay construction. All nodes form a locality aware signalling mesh at first, before the overlay is built. For the tree construction every node keeps distance information of all other participating nodes and probes their available bandwidth. Narada in effect has a linear space- and a quadratic message complexity. It works very well for small groups but does not scale to higher numbers of participating nodes. BCBS with a considerably lower protocol overhead on the other hand scales to large group sizes, as all decisions are made from local information only. The multisource approach additionally leads to a better and more equal load balancing between the nodes.

V. Conclusion and Further Work

In this paper we presented BCBS, a multi source load balancing algorithm for live streaming scenarios. The described algorithm is fair in the sense that it leads to an equal distribution of the load to all participating nodes. It creates network efficient streaming overlays using a very low protocol overhead. The resulting streaming network is stable, as meshes are created with almost all nodes being served by more than one source node.

In our future work, we plan to investigate possible improvements. So far, BCBS creates the overlay on the events of joining or departing nodes only. With periodical optimisation it should be possible to further decrease the stress and stretch, as with a higher distribution of participants the length of the paths through the backbone can be reduced. Introspection of the packets could be introduced in order to achieve a higher quality at the receivers. The knowledge of structure especially of the video data could be used to schedule more significant packets to be sent by more reliable nodes and vice versa.

Another important question is how stable BCBS performs under a high node failure rate or in case of attacks. Currently we plan to investigate this question in an analytical study and by implementing a failure model for the nodes in our simulation model and examining the resulting packet loss rate. In addition, we plan to explore how many nodes or links an attacker would have to take out in order to cause a fragmentation of the overlay.

References

[1] S. Deering, “Rfc 1112: Host extensions for ip multicasting.” [Online]. Available: http://faqs.org/rfcs/rfc1112.html

[2] “akamai.” [Online]. Available: http://www.akamai.com/

[3] Y.-H. Chu, S. G. Rao, and H. Zhang, “A case for end system multicast,” in *Measurement and Modeling of Computer Systems*, 2000, pp. 1–12.

[4] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: A Decentralized Network Coordinate System,” in *Proceedings of the ACM SIGCOMM’04*, 2004.

[5] T. Strufe, “Ilmstream: Efficient multimedia streaming in decentralised distributed systems,” in *Proceedings of The World Automation Congress*, 2004.

[6] M. Costa, M. Castro, A. Rowstron, and P. Key, “PIC: Practical Internet coordinates for distance estimation,” March 2004.

[7] A. Vargas, “Omnet++.” [Online]. Available: http://www.omnetpp.org/

[8] A. Medina, A. Lakhina, I. Matta, and J. Byers, “BRITE: Universal topology generation from a user’s perspective,” Boston University, Tech. Rep. 2001-003, January 2001.

[9] T. Bu and D. Towsley, “On distinguishing between Internet power law topology generators,” in *Proceedings of INFOCOM*, 2002.

[10] A. Shimbel, “Structural parameters of communication networks,” *Bulletin of Mathematical Biophysics*, vol. 15, pp. 501–507, 1953.

[11] S. Milgram, “The small world problem,” *Psychology Today*, pp. 60–67, May 1967.

[12] M. Hefeeda, A. Habib, B. Boyan, D. Xu, and B. Bhargava, “Promise: peer-to-peer media streaming using collectcast,” Purdue University, Tech. Rep., August 2003.

[13] W. Zhao and S. K. Tripathi, “Bandwidth-efficient continuous media streaming through optimal multiplexing,” in *Proceedings of the 1999 ACM SIGMETRICS Measurement and modeling of computer systems*, vol. 27, no. 1, May 1999.

[14] R. Rejaie and A. Ortega, “PALS: Peer-to-Peer Adaptive Layered Streaming,” in *Proceedings of the ACM NOSSDAV ’03*, June 2003.

[15] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, “Distributing streaming media content using cooperative networking,” in *Proceedings of ACM/IEEE NOSSDAV*, May 2002.

[16] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “SplitStream: High-bandwidth content distribution in a cooperative environment,” in *Proceedings of (IPTPS’03)*, 2003. [Online]. Available: citeseer.ist.psu.edu/castro03splitstream.html

[17] C. Otto, “Effizientes Scheduling in kooperativen Multimedia-Streaming-Systemen,” Master’s thesis, TU Ilmenau, 2005.