A VoD System Model Based on BC Graphs

Xiaoqiang Chen[a], Jianxi Fan[*,b], Cheng-Kuan Lin[c], Baolei Cheng[d] and Zhao Liu[e]

School of Computer Science and Technology, Soochow University, Suzhou 215006, China

{[a]1127404010, [b]jxfan, [c]cklin, [d]chengbaolei, [e]liuzhao}@suda.edu.cn

Keywords: BC-VoD model, independent spanning trees, content distribution, P2P network.

Abstract. BC graphs are important interconnection networks, which include hypercubes, crossed cubes and other hypercube's variants. In this paper, we propose a new model, named BC-VoD model, to approach VoD system, which utilizes BC graphs. At the same time, we use independent spanning trees for content distribution. The simulation results show that this model has advantageous performance.

1. Introduction

With the rapid development of the Internet, VoD (Video on Demand) system and Live steaming system gradually become the core technology of websites, such as youtube.com, twitch.tv, and so on. One of the major challenges faced by multimedia streaming services is serving a massive number of concurrent users online. Accordingly, Peer-to-Peer (P2P) has been a better way, uses up all nodes' bandwidth, storage and other resources in the system and provides services to other clients, which reduces the dependence on server. Nowadays, the researches of P2P VoD system mainly are focused on reducing the latency and the effect of frequent entering and leaving of nodes [1], [2].

Many studies show that the vast majority of users' requests are concentrated in several popular movies due to the inherent characteristics of human behavior. Thus people can use IP Multicast technology to design VoD system for saving network bandwidth and increasing system capacity. However, this technique is restricted by backbone, and it can only be applied to the LAN [3]. Therefore, more and more people start turning to perform P2P on VoD system. Recent research results on P2P streaming generally take three classifications [4] as Single-Tree, Balanced Multi-Tree, and Mesh-Based system. Fouliras came up with a protocol called LEMP [5], which is based upon a control hierarchy of successive levels for the clients. Do et al. proposed P2VoD and introduced the concept of generation and a novel caching scheme to deal with failures effectively [6]. Chow et al. constructed the tree topology model rooted at the source server, nodes in the tree is a cluster actually [7]. Nodes in the cluster can be structured as multi-tree or hypercube topologies.

The classic structure of P2P network Tapestry is a hypercube structure. Recently, [8] adopted the crossed cube into the design of the structure of P2P network. Then, we can further study interconnection network of BC graphs [9] which include hypercubes, crossed cubes, Möbius cube, and twisted cube, etc. In the other way, Yang et al. [10] gave the algorithm which can construct n ISTs in n-dimensional hypercube. Furthermore, we have solved the constuction of ISTs on a class of BC networks include hypercubes, crossed cubes, locally twisted cubes, Möbius cubes, and et al. [11]. Thus, we will use excellent properties of ISTs for content distribution in our model, which enable more fault-tolerance than the other systems in [12], [13]. In this paper, we need to build a practical topology and an efficient content of distributed scheme, so we will propose a new model—BC-VoD model. To guarantee good Quality of Service (QoS), at the same time, we enable BC-VoD model to use BC graphs and independent spanning trees (ISTs for short) to make network run smoothly, so that users will have better experience without delay.

2. Architecture of BC-VoD

In this section, we will give the formal definition of our BC-VoD model and its fundamental rules. We follow the definition of BC graphs from [9] and the definitions of node-disjoint and Independent

spanning trees(ISTs) from [14]. Now we first introduce some preliminaries.

A. Preliminaries

Table 1 shows some notations of BC-VoD.

Table 1. Notations of BC-VoD

Notation	Meaning	Notation	Meaning
tw_x	the time window of model.	l_i	the i-th layer in network.
pos_x	the position address of a layer.	RN	the root source server of network.
x_i	a node in layer.	LR_i	the first node of i-th level.

B. BC-VoD Structure

Definition 1. In BC-VoD model, first we deploy a *Root Node RN* as our resource server which contains all the original resource of our network, and manage the topology of network model. Then we group all peers into *i* levels, the upper level provide the resource to lower level. Each level has a time window setting, and a number of node settings. Those two settings are determined by the real system environment.

Fig. 1 shows the multi-tier topology that defined by *Definition 1*.

Definition 2. Within level l_i, all peers are organized in a multi-tier BC graph topology. For instance, it can be a Möbius cube, a locally twisted cube or other cube in the family of BC graph. The layer *n* is a *n*+3 dimensional BC graph, where $n \in \{0, 1, 2,..., maxlayer\}$, and the next layer is *n*+3+1 dimensional BC graph. The *maxlayer* number and the number of nodes setting *num* have this relationship:

$$(maxlayer+1) * 8 = \text{num} \quad (1)$$

Definition 3. Each node x_i in particular level has address like this:

$$\underbrace{(t_1 t_2 ... t_{\max}}_{tw_x} \underbrace{x_1 x_2 ... x_m)}_{pos_x} \quad (2)$$

The meanings of tw_x and pos_x are given in the preliminaries part. The connection rule between two nodes in the same level of the same layer is according to the definition of specific BC graphs. Between layer l_n and layer l_{n+1}, assume x_i is a node in the l_n and x_j is a node in the l_{n+1}, if the last *n* bit in x_j's address is equal to x_i's address, then they have a connection. For example, (0000, 00000) and (0001, 00001). As discussed above, the first layer in every level is a three dimensional BC graph, we denote the node *zero node* (its *pos* bit is all zero) in the first layer as the the *Local Representative* LR_i).

Algorithm 1 shows how to calculate the node address and Fig. 2 shows the inner level topology that based on hypercube, which helps to show *Definition 2* and *Definition 3*.

Definition 4. In every layer (except the first layer), some nodes will have the direct connection with upper layer. They will be the first ones to obtain resources. So if those node have largest upload bandwidth in the layer, other nodes will be faster to get resources from them.

By *Definition 2*, when we manager our network, in *n*-th layer, we need to sort the nodes by their upload bandwidth, and select the top 2^{n+1} nodes. We call the nodes as *Backbone nodes*. When network state changed, the root node should redo this process.

C. Node Joining

Using our model, the join process of new node follows the five steps: (1) The new node x_{new} sends a *join request* message to *RN*, which contains the IP address and the video id that it wants. (2) The *RN* receives the message. If video is valid or the number of nodes exceeds the limit, the *RN* will deny the request. Once *RN* accept the request, *RN* sends a message to the valid neighbor nodes next to x_{new}. (3) x_{new} sends the *ICMP* package to all neighbor nodes. After that, it sends the result to *RN*. (4) *RN* runs below algorithm and sends address of x_{new} in our network. (5) *RN* sends control message to all neighbors of x_{new}, let them know that x_{new} has joined the network, so they can provide resource to x_{new}.

Algorithm 1 Node Address Calculation

Input: result list from x_{new},timestamp of now t_n, the timewindow range of now t_{range}, the t_w of current;

Output: the network address of x_{new};

Begin

1: sort the result list l_i from x_{new} in ascending order;
2: **if** t_n in t_{range} **then**
3: get the usable pos_x of current level and append it into a list l_{ad};
4: **for** each w in l_{ad} **do**
5: calculate the neighbors of w into a list l_{nei};
6: $sum[w] \leftarrow 0$;
7: **for** each v in l_{nei} **do**
8: **if** $l_i[v]$ exist **then**
9: $sum[w] \leftarrow sum[w] + l_i[v]$;
10: **else**
11: $sum[w] \leftarrow sum[w] + \max(l_i)$;
12: **end if**
13: **end for**
14: **end for**
15: the pos_x of node is min(sum);
16: **else if** the pos_x is 000
17: $tw_x \leftarrow tw_x + 1$;
18: **end if**
19: combine tw_x and pos_x into the network address of node;

End

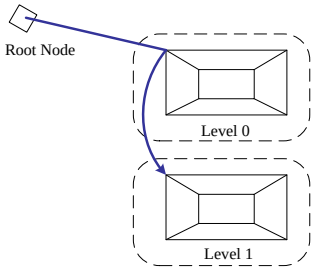

Fig. 1. The multi-tier BC graph topology.

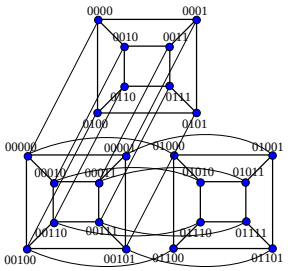

Fig. 2. The inner level topology.

D. Node Aggregation Query

In content distribution part, we will give an active routing method, in which we need to know who has the maximum bandwidth in a layer, BAWA have given the description of Node Aggregation Problem [15] (The Node Aggregation Problem: Devise a scheme to enable any node in a P2P network to issue a query that computes an aggregate function (MIN, MAX, COUNT, SUM, AVG) over data residing at nodes in the network). Under our situation, the aggregate function should be maximum bandwidth function and we only need to query the result in one layer at the same time. Then we give two algorithms. The network environment can determine which algorithm to use.

The Server Based Algorithm: When a node x in layer l_i issues a query, it should send a query message to *RN*. *RN* receives the message and sends a control message to all nodes in l_{i-1} layer. Then all the nodes in l_{i-1} send its bandwidth information to *RN*. *RN* selects the node which has maximum bandwidth, send a control message to it and ask it to provide resource to x.

The Node Based Algorithm: When a node x in layer l_i issues a query, it sends a query message to *LR* in layer l_{i-1}, Then *LR* sends its bandwidth value and its address to all its neighbors. The neighbor that receives a value checks if value more than its own bandwidth value. If so, the neighbor resends its bandwidth value to all its neighbors. Each node stops 2*(2**maxlayer* + 1) time steps after it first receives bandwidth value. When stopped, if a node find its bandwidth is the biggest one, it should connects to the querying node and provides the resource to the querying node.

E. Nodes Departure

When a node x_{old} wishes to leave the network, it sends a *quit message* to *RN*. *RN* receives the message and judges if x_{old} is a Backbone node. We have two cases. Firstly, it's not a Backbone node, then *RN* controls the last joined node x_{last} in this level to replace the position of x_{old}. Secondly, after rebuilding the connection of x_{last}, x_{old} is a Backbone node. Then *RN* also needs to replace the x_{old} position with the last joined node. After that, *RN* should redo the Backbone node building process.

F. Failure Detection

In the model design, every node should send heartbeat package to *RN* and its neighbor periodically. Once *RN* can not receive the heartbeat package from node x_i, *RN* needs to send a control message to all neighbors of x_i and let them send ICMP package to detect the state of x_i.If all neighbor nodes can not reach x_i, we say x_i is a failure, then we will do the same thing like node departure. Otherwise, *RN* should let all the neighbor node keep watching the state of x_i until *RN* can reach x_i.

3. Content Distribution

For a P2P VoD system, it has two main play behaviors, one is on-demand, another is order to play. From the user's point of view, during his watch, he is very likely to take some VCR actions, look back or fast forward. When such actions are processed, if a node still contains resources from the level where it belong to, there might cause a long buffering time. Our solution is to contain resources from other upper level. Then the node will build a connection with a *LR* in upper level, and direct contain resource from it. According to the definition of BC-VoD and actual demand. We should design three content distribution ways: routing between levels, routing between layers, and routing inner layers. In the detail, the node work in two modes, *active mode* and *passive mode*. When users watch video in order, the passive mode is on. When user takes VCR action, the active mode is on.

A. Routing between levels

Node is work in passive mode. Only Backbone nodes are involved in this process. As long as the upper level is full, the lower level will be established. The upper level *LR* will catch enough data and provide the resource to next level *LR*. Then the lower level *LR* can distribute content in its level.

B. Routing between layers

In this way, node is working in passive mode. According to the definition, we can know that there are 2^{n+1} connections between two layers, the lower layer Backbone node will have direct connection with the upper layer, so they can contain the resource that is just one hop from upper layer. After backbone node has catched resource, it distribute content in its layer.

C. Routing inner layers

An important advantage of our model is that we will use the excellent properties of multiple ISTs [11] for inner layer content distribution. Now, we introduce the concept of unit cube. Three dimensional BC graph only includes hypercube and crossed cube of three dimensional and any other hypercube variants with $n(n>3)$ dimension can be constructed by them, so we call them unit cube [9].

Based on the results in [11] and [10], the ISTs of three dimensional hypercube and crossed cube have the isomorphic structure. For routing inner layers, in first layer, it only have one node (*LR* node) which contains resource from *LR* or upper level, so we only construct ISTs rooted at *LR* node. In n ($n > 0$) layer, we construct ISTs rooted at an every Backbone node. Then, those nodes use the ISTs to distribute content to all leaf nodes. Observing these ISTs, there are three node-disjoint paths available from node 0 to any other node. Finally, all nodes in the layer have the resource what they need.

D. Active routing

When user takes VCR operations such as fast-forward and rewind, he may need the chunk which he doesn't have especially with fast-forward operation. In our model, the lack resources will be requested from the upper layer to use active routing. First, node will send a request to upper layer *LR* or server (depend on which algorithm is used), and follow the algorithm to get the resource. Through this way, we can easily make sure that the node will receive chunks what it wanted.

4. Performance Evaluation

So far, we have given all the details about our model. Now, we should do some simple analysis to show the advantage of combining BC graph with multiple ISTs and some influence factors.

A. Bandwidth utilization

In multi-tree model, each node has multiple roles in the network. It can be the central node in one tree or be leaf node in other trees. Compared to the other single tree model, the upload and download bandwidth of each node have been fully used. At the same time, the backbone node also undertakes

the task of transferring the content to lower layer. This means that each node has enough workload, any bandwidth allowance will not be wasted. Thus, We have a clear description of the bandwidth performance of our model, our model can effectively improve bandwidth utilization.

B. Richness of resources

It is obvious that the more resources each node obtained, the video will play more smoothly. In a real VoD system, each node will cache the chunk in a buffer, in which it should always have next desired chunk, if not, there must cause a pause. In previous work, they all use multiple trees to distribute content, and they all proved that this solution has a good performance. The main reason is that multiple trees can guarantee the rich in resources. Correspondingly, our model uses ISTs(Paths between root and any other node in different ISTs are node-disjoint) to distribute content. Different chunks can be simultaneously transmitted within layer. Once a node lack of appropriate chunk, it can turn on the active mode and request resource from another unit cube within same level or other level. Therefore, compared to multi-tree model, the BC-VoD model has better richness of resources.

C. Simulation

Next, we conduct simulations by locally twisted cube, Möbius cube, crossed cube, and hypercube to evaluate the performance of BC-VoD model. We use the two performance metrics:(1) The Average delay: This metric measures the average delay of chunk dissemination. (2) The Worst delay: This metric measures the worst delay of chunk dissemination.

During the simulation, the video streaming rate is regarded as unit bandwidth, and the single chunk is regarded as 1. Referring to other researcher's work, we also let our network be heterogeneous. We both consider transmission delay and propagation delay. For transmission delay, it obeys the truncated exponential distribution. The mean of this distribution is 8, and the lower and upper bound is 1 and 24, respectively. For the propagation delay, it obeys the truncated normal distribution. The mean of this distribution is 8, the variance is 4, and the lower and upper bound are 1 and 16, respectively.

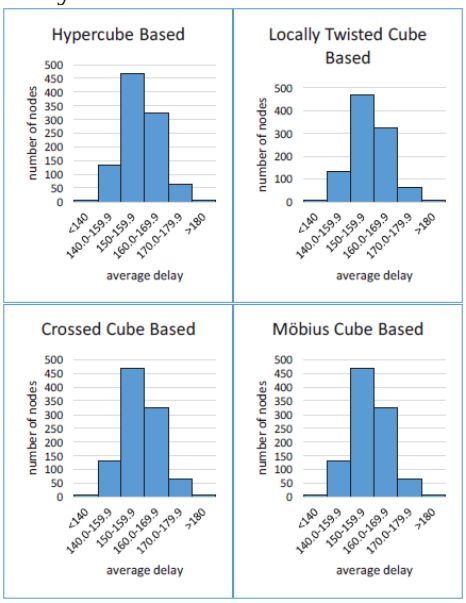

Fig. 3. The average delay with different topologies.

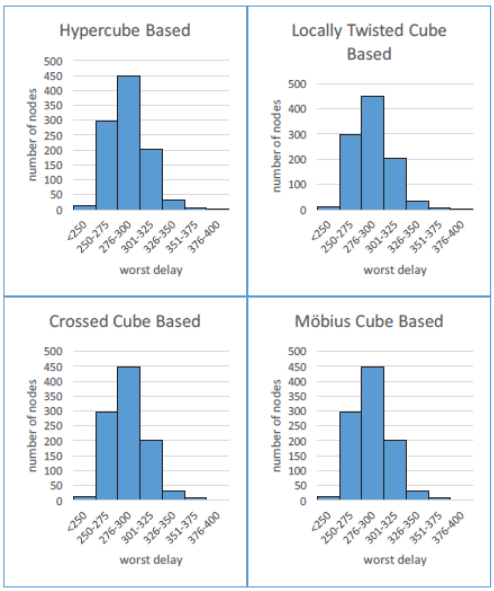

Fig. 4. The worst delay with different topologies.

Fig. 3 shows the average delay with 1000 chunks in 4 different topologies. There is no significant difference between the average delay of them.This phenomenon fits our model specification. Fig. 4 shows the worst delay with 1000 chunks in 4 different topologies. There is no significant difference between the worst delay of them. This phenomenon also fits our model specification.

Fig. 5 and Fig. 6 plot the time and communication costs of node aggregation process. The *X*-axis depicts the number of layers in one level. The *Y*-axis is the costs. Fig. 5 is the time steps and Fig. 6 is messages that sent in whole node aggregation process. The hypercube was used in this simulation. The point in the graph is the mean value of 100 times simulation. From the result, we can tell that the cost of our node aggregation algorithm is in a reasonable orientation. But with the increase of the

number of layers, the communication cost is raising very rapidly. So the number of layers settings must be set very carefully. In another hand, the time cost is related to the network diameter in one level, and time steps shows linearity relation with the number of layers.

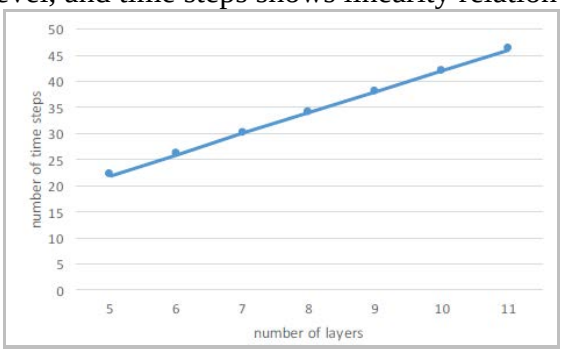

Fig. 5. Time cost of Node Aggregation Query.

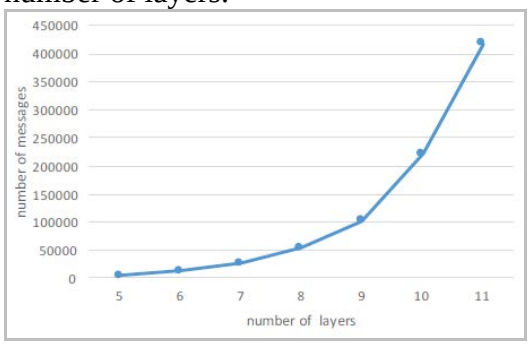

Fig. 6. The number of messages in Node Aggregation Query.

Fig. 7 shows the average routing delay of BC-VoD and HyperCuP. From [16], we known that the HyperCuP's network scale depends on the real network environment. So when we use HyperCuP to support media steaming, it's very likely to have a bigger network diameter than our BC-VoD. And the network diameter is the most important impact of average routing delay. BC-VoD is better than HyperCuP in average routing delay, see Fig. 8. Fig. 8 shows the average costs of node joining and departure. In BC-VoD, when a node joins or leaves network, it needs to communicate with *RN*, but the HyperCuP doesn't need those cost. So view from this aspect, it takes some cost. The pervious work [12] is a multitree structure, where the whole content will broken down to some stripes. Clearly, the connectivity of multitree is worse than BC graph, so when a node is fault and is not recovered immediately, one of those strips are likely to loss in the network and need source server to resend it. BC-VoD does not have the above problem. But in the other hand, our model can not have the same good transmission efficiency. In [11], we have proposed the definition of conditional BC networks which include hypercubes, crossed cubes, locally twisted cubes, Möbius cubes, and et al. and presented a general algorithm to construct *n* isomorphic ISTs in any *n*-dimensional conditional BC network. Comparing to the ISTs in Möbius cubes [14], they all have the isomorphic structure. Then, we can infer that the VoD System Model based on any conditional BC network has the similar performance to that on Möbius cube with the same dimension.

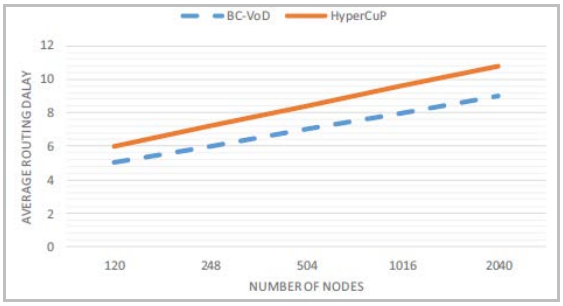

Fig. 7. Average routing delay.

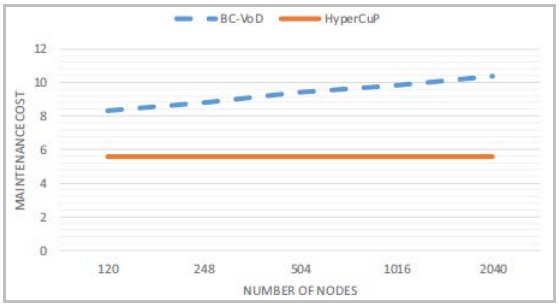

Fig. 8. Maintenance cost.

5. Conclusion

In this article, we have proposed BC-VoD model, which puts BC graphs and ISTs into P2P VoD service. BC graphs can bring us the benefits of low latency and ISTs can be used to achieve a good content distribution scheme. Detailed simulations under different kinds of BC graphs are performed, which showed that the worst delay and average delay of content distribution are low. The time cost and communication of node aggregation algorithm is better than previous work [15]. We can also obtain similar results on conditional BC graphs. At the same time, our model is robust for node joining and departure, and the network topology is scalable.

6. Acknowledgment

This work is supported by National Natural Science Foundation of China (No. 61170021 and No. 61572337), Natural Science Foundation of Jiangsu Province (No. BK2012624), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 14KJB520034), the Jiangsu Planned Projects for Postdoctoral Research Funds (No. 1501089B) and China Scholarship Council Fund (No. 201308320114).

7. References

[1] C. M. Cheng, S. L. Tsao: Adaptive Lookup Protocol for Two-Tier VANET/P2P Information Retrieval Services. IEEE Transactions on Vehicular Technology 64(3), 1051-1064 (2015)

[2] J. Sun, Y. Zhou, Y. Duan, Z. Guo: A Low-Latency Peer-to-Peer Live and VOD Streaming System Based on Scalable Video Coding. In Visual Communications and Image Processing Conference, 2014 IEEE 319-319 (2014)

[3] C. Zheng, X. Wang, J. Zhao, X. Xue: P2P Video-on-Demand Content Distribuiton Schemes. Journal of Software 11, 2942-2954 (2007)

[4] Y. Liu: Delay Bounds of Chunk-Based Peer-to-Peer Video Streaming, IEEE/ACM Transactions on Networking 18(4), 1195-1206 (2010)

[5] P. Fouliras, S. Xanthos, N. Tsantalis, and A. Manitsaris: LEMP: Lightweight Efficient Multicast Protocol for Video on Demand, ACM Symposium on Applied Computing 2, 1226-1231 (2004)

[6] T. T. Do, K. A. Hua, and M. A. Tantaoui: P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment. IEEE Communications Society 3, 1467-1472 (2004)

[7] A. L. H. Chow, L. Golubchik, S. Khuller, Y. Yao: Performance Tradeoffs in Structured Peer-to-Peer Streaming. J. Parallel Distrib.Comput 72, 323-337 (2011)

[8] H. Shen, J. Fan, B. Cheng, and C. Lin: A Model Based on Crossed Cubes for VoD Services. International Conference on Systems and Informatics 659-664 (2014)

[9] J. Fan, and X. Lin: The *t*/*k*-Diagnosability of the BC Graphs. IEEE Transactions on Computers 54(2), 176-184 (2005)

[10] J. Yang, S. Tang, J. Chang, and Y. Wang: Parallel Construction of Optimal Independent Spanning Trees on Hypercubes. Parallel Computing 33(1), 73-79 (2007)

[11] B. Cheng, J. Fan, and X. Jia: Dimensional-Permutation-Based Independent Spanning Trees in Bijective Connection Networks. IEEE Transactions on Parallel and Distributed Systems 26(1), 45-53 (2015)

[12] A. Nicolosi and S. Annapureddy: P2PCast: A Peer-to-Peer Multicast Scheme for Streaming data. IRIS Student Workshop, MIT 1-13 (2003)

[13] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl: HyperCuP — Hypercubes, Ontologies, and Efficient Search on Peer-to-Peer Networks. Lecture Notes in Computer Science, 2530, 133-134 (2003)

[14] B. Cheng, J. Fan, X. Jia, S. Zhang, and B. Chen: Constructive Algorithm of Independent Spanning Trees on Möbius Cubes. The Computer Journal 56(11), 1347-1362 (2013)

[15] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani: Estimating Aggregates on a Peer-to-Peer Network, Technical Report, Stanford: Stanford University 1-13 (2003)

[16] S. Mario, S. Michael, D. Stefan, N. Wolfgang: HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. Springer Berlin Heidelberg, 112-124 (2003)