Tree Network Coding for Peer-to-Peer Networks

Arne Vater Christian Schindelhauer* Christian Ortolf

Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 51
Freiburg im Breisgau, Germany
{vater, schindel, ortolf}@informatik.uni-freiburg.de

ABSTRACT

Partitioning is the dominant technique to transmit large files in peer-to-peer networks. A peer can redistribute each part immediately after its download. BitTorrent combines this approach with incentives for uploads and has thereby become the most successful peer-to-peer network. However, BitTorrent fails if files are unpopular and are distributed by irregularly participating peers. It is known that Network Coding always provides the optimal data distribution, referred as optimal performance. Yet, for encoding or decoding a single code block the whole file must be read and users are not willing to read $O(n^2)$ data blocks from hard disk for sending n message blocks. We call this the disk read/write complexity of an encoding.

It is an open question whether fast network coding schemes exist. In this paper we present a solution for simple communication patterns. Here, in a round model each peer can send a limited amount of messages to other peers. We define the depth of this directed acyclic communication graph as the maximum path length (not counting the rounds). In our online model each peer knows the bandwidth of its communication links for the current round, but neither the existence nor the weight of links in future rounds.

In this paper we analyze BitTorrent, Network Coding, Tree Coding, and Tree Network Coding. We show that the average encoding and decoding complexity of Tree Coding is bounded by $O(kn \log^2 n)$ disk read/write-operations where k is the number of trees and n the number of data blocks.

Tree Coding has perfect performance in communication networks of depth two with a disk read/write complexity of $O(pnt \log^3 n)$ where p is the number of peers, t is the number of rounds, and n is the number of data blocks. For arbitrary networks Tree Coding performs optimally using $2(\delta+1)^{t-1} p \log^2 n$ trees which results in a read/write complexity of $O((\delta+1)^{t-1} n \log^3 n)$ for t rounds and in-degree δ.

*Partly supported by DFG research fund Schi 372/5-1.

SPAA'10, June 13–15, 2010, Thira, Santorini, Greece.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems — *Distributed applications*;
E.4 [**Data**]: Coding and Information Theory — *Nonsecret encoding schemes*;
F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity — *Nonnumerical Algorithms and Problems*

General Terms

Algorithms, Performance

Keywords

Peer-to-Peer Networks, BitTorrent, Network Coding

1. INTRODUCTION

The exchange of data without centralized infrastructure is the main motivation for the wide-spread use of peer-to-peer networks. From a user's perspective the fast distribution of large files is the killer argument to choose peer-to-peer network software. For such a task the IP Multicast protocol seems to be the best solution, allowing routers to duplicate packets on their paths to their destination, thus relieving the bottleneck at the server [17, 6]. However, IP Multicast suffers from the absence of reliable delivery and the lack of support of most Internet service providers.

Peer-to-Peer Networks.

Peer-to-Peer Networks started in 1999 with Napster and Gnutella which swiftly became very successful although they were not very elaborated. In the following years, researcher focused on finding robust network structures and efficient lookup services, like CAN [13], Chord [16], Pastry [15], and Tapestry [8]. Later on, for some of these networks, efficient multicast extensions were proposed, e.g. Bayeux [19], CAN-Multicast [14], and Scribe [3], filling the gap of the unsupported multicast in the Internet network layer.

In a multicast tree the leaf position is the most favorable one, since they do not upload any data to others. Usually, the upload is the crucial bottleneck in peer-to-peer networks, since asymmetric connections designed for client-server networks provide larger download than upload capacities (not mentioning the legal distinction between uploaders and downloaders). A solution was presented with Splitstream [2], where files are partitioned into small blocks, such that a peer can start redistributing blocks immediately

after downloading while continuing receiving further blocks of the same file. The resulting multiple distribution trees are overlaid to achieve fairness among peers by balancing both upload and download for each of them.

BitTorrent [5] incorporates the block-based approach of Splitstream and combines it with incentives for uploading blocks. So, BitTorrent has become traffic-wise the most successful peer-to-peer network.

Network Coding.

In their seminal paper Ahlswede et al. [1] showed that the encoding of data can improve the data throughput beyond the limits of standard packet delivery. Later work showed that linear combinations of data units achieve optimal network information flow. This idea has been adapted for use in peer-to-peer networks like Practical Network Coding [4]. Some peer-to-peer networks use this new method. Yet, BitTorrent without any coding technique remains unchallenged.

In our opinion the main obstacle for the wide-spread use of Network Coding is the high computational complexity of the coding and encoding process. For encoding (or decoding) a single code block the whole file must be read and users are not willing to read $O(n^2)$ data blocks from hard disk for sending n message blocks. The additional overhead after downloading the code blocks is simply not acceptable for most users: When a user downloads 4 GByte of data consisting of 1024 blocks, Network Coding requires disk operations reading 4096 GByte on each participating host while BitTorrent only reads 4 GByte of data.

Our goal is to find coding schemes which provide the same information flow as Network Coding with a disk access complexity comparable to BitTorrent.

Previous Work.

In our previous work we have presented new network codes with small read/write complexity. In [11] we have introduced Pair Coding. We have shown that it performs at least as good as BitTorrent with a constant factor increase in complexity. For some scenarios it outperforms BitTorrent, while Pair Coding fails for many scenarios like BitTorrent.

Furthermore, we have introduced Tree Coding [10]. It outperforms a special version of Pair Coding called Fixed Pair Coding, while its relationship to Pair Coding is unknown. Pair Coding and Fixed Pair Coding are as efficient as BitTorrent. Likewise the complexity of Tree Coding for $k > 1$ trees was stated as an open problem, which is solved in this paper.

Coding Model.

All coding schemes presented here are restricted versions of the Practical Network Coding introduced in [4]. A large file of length m over an alphabet Σ (e.g. binary alphabet, bytes, words) is partitioned into n equal units of size $s = \lceil \frac{m}{n} \rceil$. We denote the blocks of the file by $x = (x_1, \ldots, x_n)$. The last block may be filled up with zeros. We assume $n = 2^a, a \in \mathbb{N}^+$ since this supports the binary presentation of data. The linear network code schemes presented use scalar products in finite fields.

For efficiency the block x_i is interpreted as a vector over a finite field, i.e. $x_i = (x_{i,1}, \ldots, x_{i,\ell}) \in GF[2^h]^\ell$ such that 2^h is larger than the number of blocks, but the product nh is smaller than the block size $2^{h\ell}$. Then the additional information of the encoding can be seen as a minor contribution to the packet size.

A **linear code block** is defined as

$$b(c) = \left(\sum_{i=1}^{n} c_i x_{i,\nu} \right)_{\nu \in \{1,\ldots,\ell\}} .$$

If n such linear code blocks $b(c) = b_1, \ldots, b_n$ with codes $c_1, \ldots, c_n$ have been collected the matrix $C_{ij} = (c_{i,j})$ gives the information for decoding since

$$x_{i,\nu} = \sum_{i=1}^{n} (C^{-1})_{i,j} b_{j,\nu} .$$

A random choice of the code variables produces an invertible matrix with probability of at least $1 - \frac{2}{2^h}$. Since the space requirements of the parameters (nh) grow linearly with h this allows exponential small failure probability. In practice such a failure would usually result in the transmission of an additional encoded block and presents a minor problem.

Communication Graph.

We model the communication of peers as a directed acyclic graph. For this we consider a round model where each participating peer P_i is multiply represented in the graph by nodes $P_{i,j}$ indicating the state of P_i in the j-th round. The directed edges are of form $(P_{i,j}, P_{k,j+1})$ and are weighted according to the number of blocks peer P_i can transmit to peer P_k in round j. We consider only the transmission of full blocks according to the linear encoding. By definition the edges $(P_{i,j}, P_{i,j+1})$ always exist and have weight n. In every round a peer may receive any number of blocks and send any number of blocks within the edge weight. The sum of all received blocks of a peer P is denoted as $n_r(P)$ and clearly we observe $n_r(P) \leq n$ if no unnecessary blocks are sent around. The number of outgoing blocks of a peer P is denoted by $n_s(P)$. Peers in round 1 have either all blocks or no blocks of the file. Peers of the first type are called seeds and the latter ones leeches. We define $n_r(P) = n$ if P is a seed.

In this paper we assume that there are no edges with zero weight and we consider only edges that lie on a directed path coming from a seed.

In the communication graph we define the in-degree of a node as the number of incoming edges of a node $P_{i,j}$ not counting the edge from the past round $P_{i,j-1}$. Since many nodes represent the same peer we redefine the depth of the graph as follows. All nodes corresponding to seeds have depth 0 in all rounds. The depth of nodes in the first round is also 0. In all other cases the depth is defined as

$$\begin{aligned} &\mathrm{depth}(P_{i,j+1}) = \\ &\quad \max \Big\{ \mathrm{depth}(P_{i,j}), \\ &\qquad 1 + \max_{k \neq i, (P_{k,j}, P_{i,j+1}) \in E(G)} \{\mathrm{depth}(P_{k,j})\} \Big\}. \end{aligned}$$

The depth of the communication graph describes the maximum number of different peers a piece of information passes through. According to this definition the number of rounds can easily exceed the depth of a communication graph.

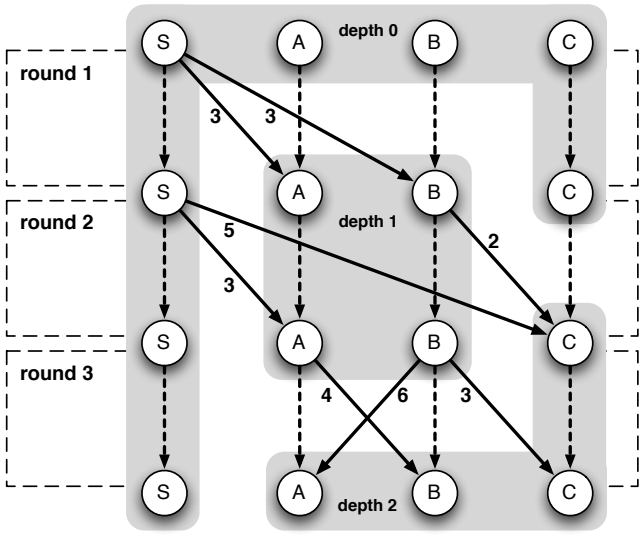

Figure 1: Communication graph.

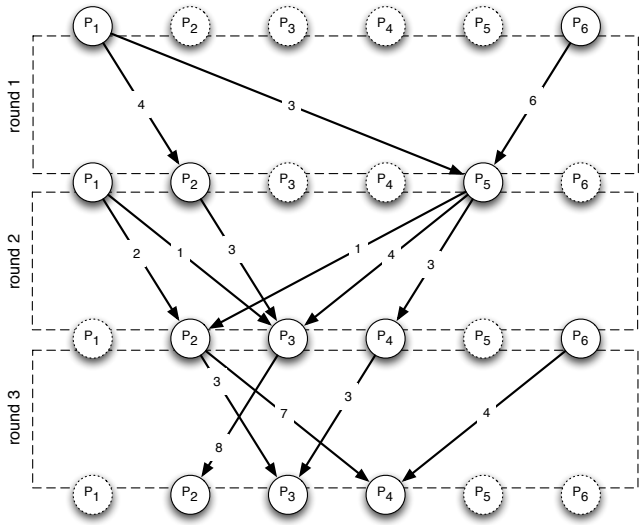

Figure 2: Example of a communication graph with 3 rounds. Inactive nodes are depicted dashed, the nodes P_1 and P_6 are seeds, and the depth of the communication graph is 3.

THEOREM 1. *[4] If there exists a flow from all seeds to a peer of size of at least n then linear network coding distributes the complete file to all these peers.*

In this paper we investigate variants of Network Coding which share this optimal performance.

Definition 1. A file sharing system S_1 **performs perfectly** if the complete file is received at all peers which have a flow of n from the set of seeding peers.

The fundamental disadvantage of Network Coding is the large computational complexity for encoding and decoding. To reflect the cost of coding we use the read/write complexity regarding disk access. We assume that any operation involving two blocks (coded or plain) can be performed in main memory, i.e. it requires only to read those two blocks and write the result. The number of blocks a peer can store in its main memory is constant (non-zero).

Definition 2. Consider a machine model where the main memory is restricted to a constant number of blocks and a mass storage device is available, which is an unbounded random access memory (hard disk). The **read/write complexity** of a peer is the number of accesses to the mass storage for an operation. It is measured with respect to the number n_r of blocks received by a peer and the number n_s of blocks sent by a peer.

We consider the worst-case given by the peer of maximum read/write complexity and the average read/write complexity which is the sum of read/write operations averaged over all peers.

Knowledge Model.

In the **offline model** the whole communication graph with all future rounds is known to all peers in advance, i.e. each node knows the activity and communication limits of all peers in all upcoming rounds. In the **online model** this knowledge is limited to the current round. Thus a plan can only be made for this round, while no information about future rounds is available. Basically, this is a guarantee that each peer remains active until the end of the current round and will be able to fulfill its transmission bounds published at the beginning of the round.

2. BITTORRENT

BitTorrent is at the moment the most popular file sharing system. Except for error detection via hash function it does not use coding. A file is divided into n blocks, that are distributed to the downloading peers. When a copy of each original block is present, the download is finished. The crucial advantage of BitTorrent compared to prior peer-to-peer systems is the capability of downloading from multiple other peers in parallel, and start uploading completed blocks before the download of the whole file is finished. The so-called policy describes the decision which block should be uploaded to whom. It is based on the current status of the whole network, including for example the progress of the receiving peer, the amount of copies of a certain block in the network, etc. BitTorrent uses incentives that encourage uploading by delivering more blocks in return. Thereby, so-called leeching, i.e. peers (leeches) download more blocks than they are willing to upload to others, is discouraged.

This game-theoretic approach of using incentives helps to increase network throughput, and together with the file partitioning this explains the success and popularity of BitTorrent. The game-theoretic aspects have been subject to a lot of research, e.g. [9, 12], and they still are. However, we focus on shortcomings of block-based file distribution systems, that are unsovable with incentives: In case a block is completely missing in the whole network, i.e. no active peer has a copy, BitTorrent is incapable of compensating this loss and no peer is able to finish the download.

Performance Analysis of BitTorrent.

In the following, we describe the communication graphs that can be handled perfectly by BitTorrent and also give the read/write complexity. We will show that BitTorrent fails for more complex communication graphs.

THEOREM 2. *BitTorrent performs perfectly for communication graphs with depth 1 for the online model and has worst case read/write complexity $n_r + n_s$ and average complexity $2n$.*

PROOF. A simple policy downloading one missing block at a time suffices. The downloading peer can explicitly request a desired block from each seed. So with each received block the information available at the downloading peer increases. Thus, the total information flow to that peer is optimal in the online model, and each block has to be sent and received exactly once, yielding the given read/write complexity in average. In worst case, a peer (in particular a seed) has to upload more than n blocks for several downloading peers, $n_s > n$, resulting in the above worst case complexity.

For the average complexity note that

$$\sum_{peer\ P} n_r(p) = \sum_{peer\ P} n_s(p) \leq np$$

where p is the number of peers. □

The following theorem shows the unsolvable shortcomings of BitTorrent. If the communication graph has a depth larger than one, even in the offline model with complete knowledge, BitTorrent cannot provide the optimal performance.

THEOREM 3. *There is a communication graph of depth 2 where BitTorrent fails in the offline model.*

PROOF. The communication graph in Figure 3 depicts an example. In round one, the seed S may transmit $n/2$ blocks to each of the peers P_1, P_2, P_3. Since there exist only n different blocks, at least two peers have an identical block after the first round. Due to the duplicate blocks at two peers, in round two (without any transmission limits), at least one of the peers P_4, P_5, P_6 cannot download all original blocks. No distribution strategy can solve this, not even for the offline model with full knowledge.

However, there exists an optimal straight-forward solution using Network Coding: In round one all transmitted blocks are linearly independent code blocks. Then in round two the nodes P_4, P_5, P_6 each can collect n of those code blocks and decode the original file, maximizing the network flow. □

3. NETWORK CODING

The problem of missing blocks can be optimally solved by using Network Coding [1] with its efficient implementation given as the Practical Network Coding scheme [4]. Instead of transmitting the original blocks of the file, linear combinations of all n blocks are distributed. Those code blocks are generated by interpreting each original block as a vector over a Galois field. If all linear coefficients used in this encoding are linearly independent, then any n code blocks are sufficient to decode all n original blocks from the file. Linear independency of the coefficients can be achieved easily with high probability by a random choice, if the order of the Galois field is chosen large enough.

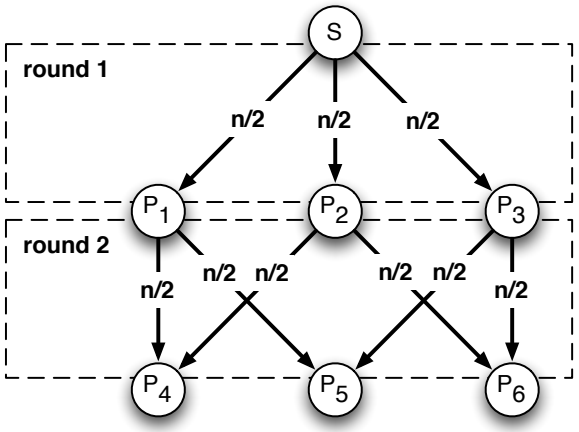

Figure 3: Scenario used in proof of Theorem 3.

THEOREM 4. *[1] Network Coding performs perfectly in any communication graph.*

However, encoding and decoding of blocks have a high computational complexity and a high read/write complexity. The computational overhead is basically the inversion of an $(n \times n)$-matrix C of the linear coefficients which has a complexity of $O(n^{2.376\ldots})$. While this is manageable with modern personal computers the read/write complexity displays the growing gap between the processing speed of the CPU and had disks.

THEOREM 5. *The read/write complexity of Network Coding of each peer is in the worst case at most $n_r \cdot n_s$ and $O(n^2)$ in the average.*

PROOF. Creating a code block requires to read n_r blocks, leading to $n_r n_s$ read operations to create n_s code blocks, which is the same for decoding like in Practical Network Coding. Since $n_r \leq n$ the average performance is at most n^2 since

$$\begin{aligned} \sum_{peer\ P} n_s(P) n_r(P) &\leq \sum_{peer\ P} n_s(P) n \\ &= \sum_{peer\ P} n_r(P) n \\ &\leq n^2 p\ , \end{aligned}$$

where p is the number of peers. □

We think that this read/write complexity is the reason that systems using Network Coding, like Avalanche [7], by far fall behind BitTorrent's success and there are empirical analyses backing up this opinion [18].

4. TREE CODING

Code blocks in Tree Coding, which was already introduced in [10], are defined by a complete binary tree. The leaves of this tree form the original file blocks multiplied by a coefficient. Starting from the second layer code blocks are generated by adding two children code blocks in the Galois field, i.e. computing the vector-wise Xor of the code blocks.

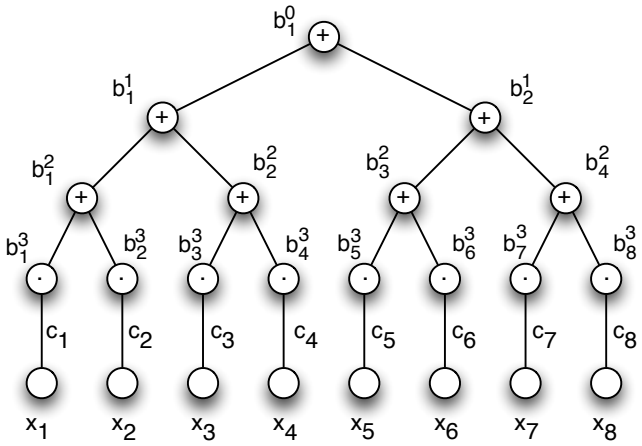

Figure 4: Coding tree with $n = 8$.

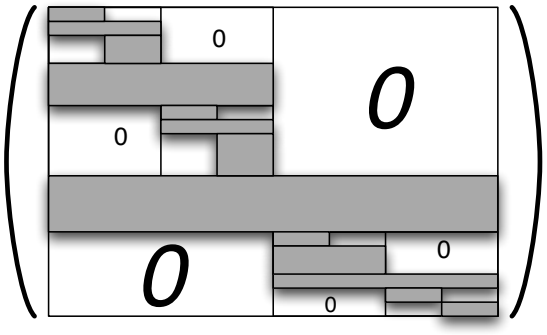

Figure 5: A typical matrix of the Tree Coding using several trees.

$$
\begin{aligned}
b_i^{\log n}(c) &= c_i x_i && \text{for } i \in \{1,\dots,n\} \\
b_i^{j-1}(c) &= b_{2i-1}^{j}(c) + b_{2i}^{j}(c) && \text{for } j \in \{1,\dots,\log n\}, \\
& && \quad i \in \{1,\dots,2^{j-1}\}
\end{aligned}
$$

Such a tree imposes an implicit partitioning of the file, which may lead to the same problems we have seen for BitTorrent, see Figure 5. The solution is to use several coding trees with different coefficients.

Decoding from multiple coding trees is computationally more complex. In the extreme case n coding trees are given by their root nodes. Then, Treecoding is equivalent to Network Coding. However, for a small number of trees the decoding remains feasible.

Definition 3. A sub-tree in a coding tree $T(c)$ is defined by the position of its root node b_i^j and contains all successor nodes. It is denoted by $T_i^j(c)$, where c is the tree's coding vector.

Definition 4. The function

$$\text{count}\left(b_i^j(c_1),\dots,b_i^j(c_k)\right) \to \{0,\dots,k\}$$

denotes the amount of blocks that are present in the coding trees $T(c_1),\dots,T(c_k)$ at the node position indicated by i and j. Furthermore, $\text{count}(T(c))$ denotes the number of code blocks from the tree $T(c)$.

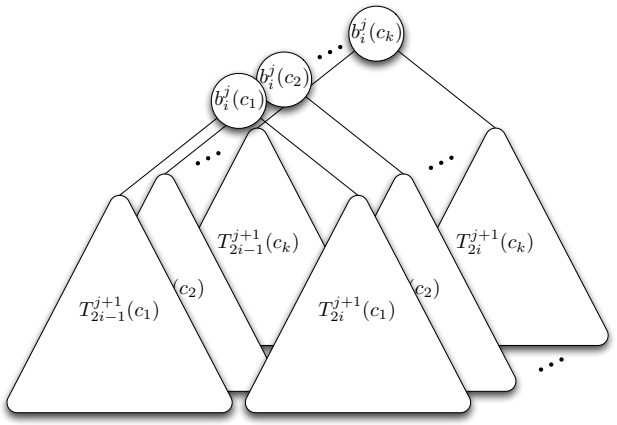

Figure 6: A set of trees $T_i^j(c_1),\dots,T_i^j(c_k)$ and their left and right sub-trees.

The following lemma will allow to abstract from the matrix representation. It shows that the rank of the trees is described by the position of the code blocks within the trees. See Figure 6 for an illustration of the notation.

LEMMA 1. *There exists a set of coding vectors $c_1,\dots,c_k$, such that for all trees $T(c_1),\dots,T(c_k)$ with*

$$\text{count}(T(c_\nu)) = \text{rank}(T(c_\nu))$$

the rank of the set of trees equals

$$
\begin{aligned}
&\text{rank}\left(T_i^j(c_1),\dots,T_i^j(c_k)\right) = \\
&\min\Big\{\frac{n}{2^j}, \text{rank}\left(T_{2i-1}^{j+1}(c_1),\dots,T_{2i-1}^{j+1}(c_k)\right) \\
&\qquad + \text{rank}\left(T_{2i}^{j+1}(c_1),\dots,T_{2i}^{j+1}(c_k)\right) \\
&\qquad + \text{count}\left(b_i^j(c_1),\dots,b_i^j(c_k)\right)\Big\}.
\end{aligned}
$$

PROOF. This can be proved by an induction over the height of the sub-trees. For leaves the statement is true, since the rank of a leaf is 1 if at least one code block exists at the leaf's position in any of the trees $T(c_1),\dots,T(c_k)$. If not, the rank is 0.

Assume that the lemma holds for depth of at least $j+1$, i.e. height smaller or equal $\log n - j - 1$. Now, consider a matrix built by the coefficient vectors of the two sub-trees and the code blocks at the current root. The matrix has the form depicted in Figure 7.

Clearly, the coefficients of the left and right sub-tree are linearly independent. It remains to prove that the coefficients $b_i^j(c_1),\dots,b_i^j(c_k)$ of the root are no linear combinations of the rest of the matrix, if the overall number of coefficients is bounded by $\frac{n}{2^j}$. If more coefficients are given, we reduce the number of root coefficients such that the overall number equals $\frac{n}{2^j}$.

First note, that within the same tree this holds since the number of block codes equals the rank of all coefficients. To ensure that it also holds for all trees we have to choose the coding vectors. These vectors are the entries of the matrix in Figure 7 at the non-zero positions. It is well known that a Vandermonde matrix fulfills this property. □

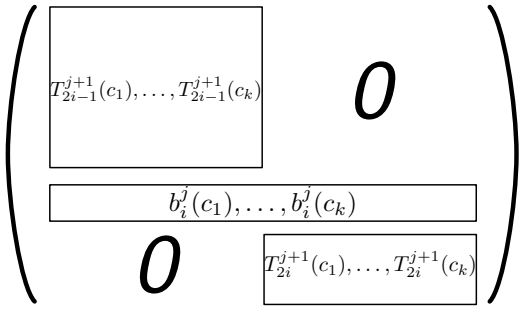

Figure 7: The resulting coefficient matrix of joining the left and right sub-trees of $T(c_1), \ldots, T(c_k)$.

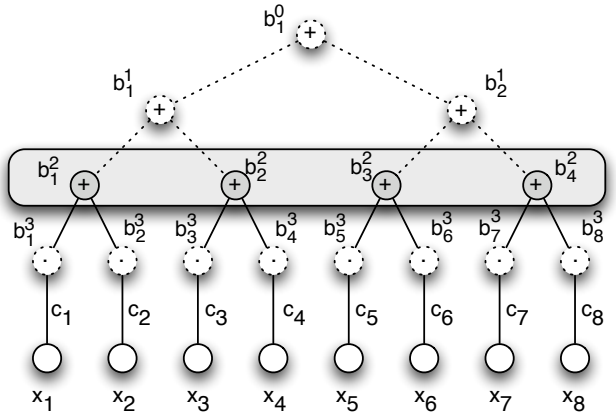

Figure 8: Complete tree level code of depth 2.

The following theorem solves the open problem stated in [10], where for the complexity of Tree Coding a bound of $O(n)$ for the case $k = 1$ has been shown. Here, we show that every strategy which relies on Tree Coding with small number of trees can be performed efficiently.

THEOREM 6. *In Tree Coding with k trees each peer has worst case read/write complexity of $O(kn \log^2 n + n_s)$ and average complexity $O(kn \log^2 n)$.*

PROOF. The seeds (peers in depth 0) calculate all k trees and store them. This takes $O(kn)$ read and write operations. Note that by definition $n_r = n$ and hence we face the complexity of $O(kn_r) + n_s$ for sending n_s code blocks in all rounds.

Each other peer follows the following strategy. The main goal is to construct all inner nodes of the k trees. The peers store code blocks in a complete binary tree T of height $\log n$. Here, in every node of depth d at most $k(d+1)$ code blocks are stored. These code blocks are linear combinations of all original blocks according to the presentation introduced for Tree Coding.

This tree T is initialized with all the received code blocks and will be updated after each round as follows: If a new code blocks has been received it will be added at the corresponding position of the tree. The other positions will be filled up with tree nodes of the same form, i.e. in sub-tree T^j_i the vectors have the form

$$(\underbrace{0, \ldots, 0}_{(i-1)n/2^j}, h_1, \ldots, h_{n/2^j}, \underbrace{0, \ldots, 0}_{n-in/2^j}).$$

After downloading the blocks of the current round the peer starts the following computation from the top of the tree to the leaves.

- Store all received code blocks into the tree at the corresponding positions.
- For $j = 1$ to $\log n$,
  for $i = 1$ to 2^j,
  for all unmarked blocks b of the parent node of T^j_i, i.e the root node $T^{j-1}_{\lfloor (i+1)/2 \rfloor}$
  - Find a vector v of form g^j_i which is linearly independent from all vectors in the sub-tree T^j_i such that it preserves the rank of the matrix of the parent tree $T^{j-1}_{\lfloor (i+1)/2 \rfloor}$ if it is replaced with b.
  - If such a vector v exists store v into the tree and mark b.

The number of disk read/write operations for computing v in depth j is bounded by $n/2^{j-1}$ because this is the maximum number of linearly independent vectors with $n/2^{j-1}$ non-zero entries. By the marking mechanism it is clear that the maximum number of $k(j+1)$ code blocks in depth d is an upper bound (in fact it is also upper-bounded by $n/2^j$). Furthermore, each entry is computed only once, which leads to the following complexity for constructing the tree:

$$\begin{aligned} \sum_{j=1}^{\log n} k(j+1)2^j \frac{n}{2^{j-1}} &= 2kn \sum_{j=1}^{\log n} (j+1) \\ &= kn(3 + \log n) \log n \end{aligned}$$

As an additional term we have to consider the number n_s of sent code blocks which are read from the data structure, resulting in the worst case read/write complexity of $O(n_s + kn \log^2 n)$ and the average read/write complexity of $O(kn \log^2 n)$. □

Note that for some policies smaller read/write complexity can be achieved. For some communication graphs it is advantageous to use only a special subset of all possible tree codes. A Tree Coding is called a **tree level coding** if for each of the k trees only codes b^j_i for one level j are used. We call a tree level coding **complete** if for each tree all or no block codes of a level are available at each peer, see Figure 8.

THEOREM 7. *Tree Coding performs perfectly for*

1. *communication graphs of depth 1 in the online model with one tree with a policy with worst case read/write complexity $n_r + n_s$.*
2. *communication graphs of depth 2 in the online model with $k = 2pt \log n$ trees where p is the number of peers in depth 1, t is the number of rounds and n the number of blocks with a policy with an average read/write complexity $O(ptn \log^3 n)$.*

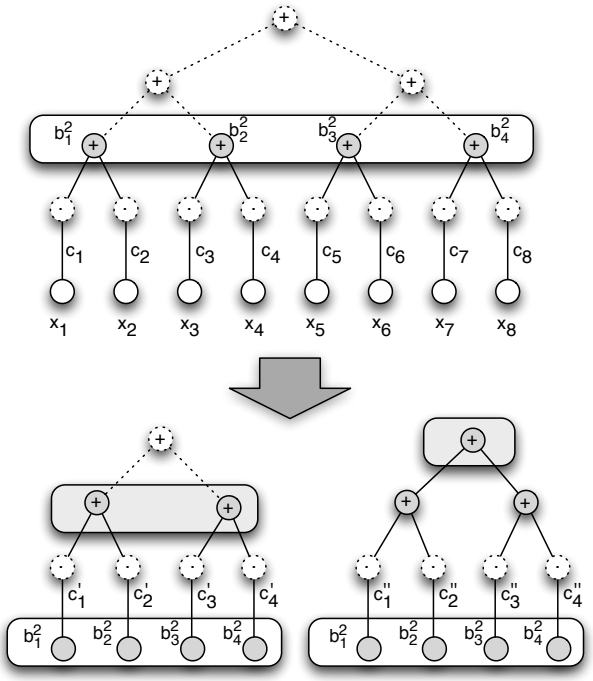

Figure 9: Recursive application of complete tree level coding generating for $w = 3$ new complete level tree codes of depth 0 and 1.

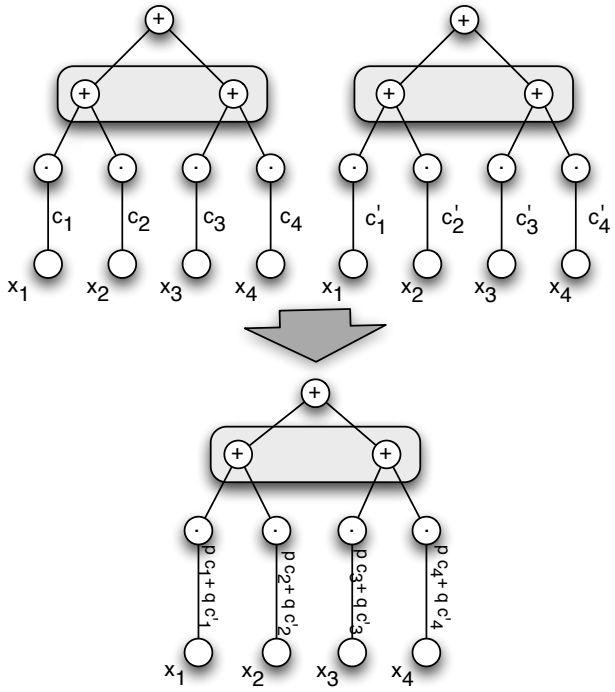

Figure 10: Tree Network Coding describes the generation of new tree codes from different existing tree codes.

PROOF. The first claim follows from the observation that BitTorrent is a degenerated version of Tree Coding and is thus implied by Theorem 2.

For the second claim we consider the following strategy. Seeds use complete tree level codes by sending in each round at most $\log n$ new trees to each peer in depth 1. In particular, if w is the bandwidth between seed S and leech P and $w = \sum_{j=0}^{\log n} w_j 2^j$ is the binary representation, then the seed S sends a new tree with a complete level of 2^j code blocks to P for all $w_j = 1$ in this round.

After t rounds a node in communication depth 1 has collected only complete levels of various depths. We denote by $m_j(P)$ the number of complete tree level codes of coding tree level j of a peer P. On request of a peer Q with communication depth 2 peer P will only send complete levels. For this, P may construct new tree codes out of a received level code. Later on, P may construct further new tree codes out of the same level for Q. Therefore, Q has to remember the amount of codes received from P for each level code.

The construction of an encoding over the code blocks of a level is done by an recursive approach of complete tree level coding. If w' is the number of blocks requested by Q for this level in this round, then tree level codes at height w'_j are generated if $w'_j = 1$ ($w' = \sum_{j=0}^{\log n} w'_j 2^j$) using given code blocks, see Figure 9.

Furthermore, we want to restrict the generation of new code blocks to two levels per round and communication partner. For this, between P and Q there is only one tree level that currently serves as basis for the generation of new tree level codes in every round. Q greedily requests as many complete unsent levels of P as possible. For the residual bandwidth, it acquires as much independent new tree level codes from the currently used new tree level code. If this level has been exhausted, then a new tree level code will be started. So, the number of new trees used in this round is bounded by $2 \log n$, since the complete level codes are just copies of existing trees built for P.

By Lemma 1 it follows that the rank of the coding matrix of Q equals the number of received block nodes if not more than n block nodes have been received. From this, the optimal performance follows.

The overall number of trees at any peer is bounded by $2pt \log n$ and the read/write complexity follows by Theorem 6. □

Up to now different trees have not been used to produce new tree codes. If we allow peers to produce new tree codes we refer to it as **Tree Network Coding**.

As an example consider two tree codes $T(c)$ and $T(c')$ in Figure 10. An intermediate node has received the tree nodes $b_1^1(c), b_2^1(c)$ and $b_1^1(c'), b_2^1(c')$. Then the node can produce some blocks of the new tree code $T(pc + qc')$, since

$$b_i^j(pc + qc') = pb_i^j(c) + qb_i^j(c') \ .$$

Clearly, it is not always possible to produce all code blocks of the new code tree from this limited input. On the other hand, re-encoding helps in the case of deeper communication graphs.

We will now prove that Tree Coding can efficiently deal with any communication graph if the number of rounds is sub-logarithmic. Again, we use complete tree level codes.

In each round new complete tree levels are generated, where the objective is to build only a small number of trees to keep the read/write complexity small.

We will only sketch the lengthy and involved proof for the optimal performance, which will appear in the full paper.

THEOREM 8. *Tree Coding performs optimally for all communication graphs in the online model with* $k = p(\delta+1)^{t-1}\log^2 n$ *trees where* p *is the number of peers in depth* 1*,* t *is the number of rounds,* δ *is the in-degree of the communication graph and* n *the number of blocks. The average read/write communication complexity is bounded by* $O(pn(\delta+1)^{t-1}\log^3 n)$.

PROOF. The following strategy is used for each node. When a peer requests information from a seed, the node receives in each round at most $\log n$ new complete tree level codes similar to the proof of Theorem 7.

If a peer Q requests w blocks from a non-seeding peer P, we consider the following cases:

1. If the request w is larger than the amount of available independent code blocks at P, then P forwards all its block nodes to Q.

2. In all other cases P produces new complete tree level codes according to the following description.

Let m_j denote the number of complete tree level codes of depth j at P. Then, the node has $\sum_{j=0}^{\log n} m_j 2^j$ code blocks. First Q calculates the number of independent complete level trees $m'_1,\ldots,m'_{\log n}$, i.e. the minimum number of tree levels which can maximize the rank of the receiver's matrix.

Now given m'_j complete tree level codes of depth j choose the largest ℓ with $\sum_{j=0}^{\ell} m'_j 2^j \le w$. Let $r = w - \sum_{j=0}^{\ell} m'_j 2^j$. Choose $s = \lfloor r/2^\ell \rfloor$ and let $q = r \bmod 2^\ell$ with binary representation $\sum_i q_i 2^i$. For the transmission the complete tree level codes $z_j = m'_j + q_j$ for depth $j < \ell$ and $z_\ell = m'_\ell + q_\ell + s$ for depth ℓ are constructed as linear combinations of all tree levels of smaller depth. See Figure 11 for an example.

Now each of the z_j new tree level codes is constructed by a linear combination of all $\sum_{\nu=j}^{\log n} m'_\nu 2^{\nu-j}$ code blocks in the corresponding sub-tree of P.

For the read/write complexity we count the number of trees after each round. By induction assume that the maximum number of different trees in each level $M = \max_i\{m_i\}$ in round t is bounded by $M(t) = 2(\delta+1)^{t-1} - 1$, which is true for the first round. In round $t+1$ each peer receives at most $\delta M(t) + \delta$ trees in each level from all sending peers. Thus the maximum entry in the next round is

$$\begin{aligned} M(t+1) &= M(t) + \delta M(t) + \delta \\ &= (M(t)+1)(\delta+1) - 1 \\ &= 2(\delta+1)^t - 1 \end{aligned}$$

proving the induction.

Since there are at most $\log n$ levels the maximum number of trees is $(2(\delta+1)^{t-1}-1)\log n$. Applying Theorem 6 proves the claimed average read/write complexity of $O(n(\delta+1)^{t-1}\log^3 n)$.

We consider a peer P_i in a fixed round t. The maximum flow $f(P_{i,t})$ in the communication graph from the seeds to $P_{i,t}$ describes the optimal information flow. In [4] it is shown that the rank of the linear vectors equals this flow when Network Coding is used. Now, we will show that Tree Network Coding achieves the same rank as the original Network Coding.

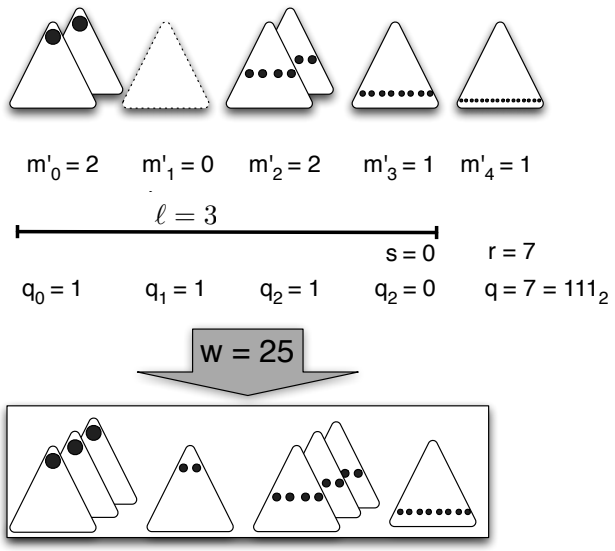

Figure 11: Dynamic generation of new tree level codes.

We prove this by an induction over a series of minimum cuts of weight $f(P_{i,t})$. First we choose the cut which separates all seeds from their successors in round 1. Then, we successively move each of the peers of the next round from one side of the cut to the other until we establish the cut which separates all peers in round 2. This way we continue until we have reached the t-th round. Clearly, each of these cuts has weight of at least $f(P_{i,t})$ since the communication graph is a directed acyclic graph, see Figure 12.

Now, we consider the series of matrices defined by all vectors which are affected by a cut and show that the rank of each of the matrices is at least $f = f(P_{i,t})$. In the first round the seeds transmit independent complete tree level codes with at least f blocks. By Lemma 1 the rank of this vector is at least f.

Assume that the rank of the matrix of all vectors on a given cut is at least f. Then, consider the vectors corresponding to the incoming edges of the node Q which will be flipped from one side to the other side of the cut. Let r be the flow which these edges contribute to the flow from the seeds to the peer $P_{i,t}$. Of course the weight of these edges must be at least r and by induction the matrix can be reduced to r code blocks on these incoming edges without decreasing the rank of the overall matrix below f. Now consider a remaining code block of the highest level. For each outgoing edge of Q with non-zero weight this code block contributes to a new encoding. So, for each possible flow to the next peer, we can find a new created code block which can replace this code block in the cut of the matrix. This argument can be repeated for the next highest code block.

Of course, the linear combinations have to be chosen with care. However, using the probabilistic method it can be shown that such rank-preserving linear combinations exist (and if the basis of the Galois field is large enough this can be guaranteed with high probability). □

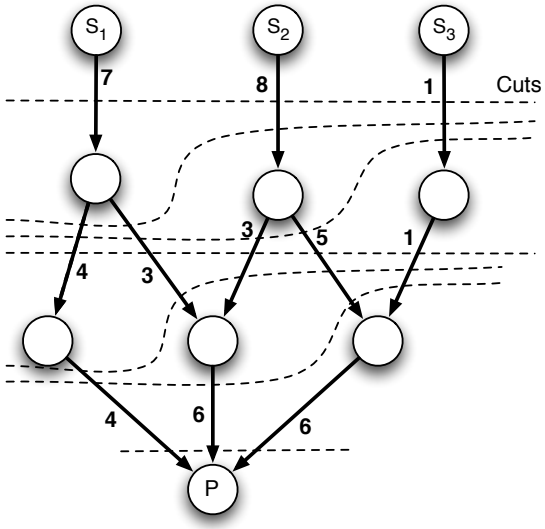

Figure 12: Series of cuts of the weight of the flow in the communication graph.

Coding Method	Depth	Average R/W Complexity
BT, TC	1	$O(n)$
TC	2	$O(ptn \log^3 n)$
TC	any	$O(p(\delta+1)^{t-1} n \log^3 n)$
NC	any	$O(n^2)$

Table 1: Upper bounds

5. SUMMARY AND OUTLOOK

In this paper we have shown that certain restricted linear network coding schemes can provide optimal performance. For this, we have presented strategies for communication networks with a restricted depth or a very small number of rounds. If the number of rounds is bounded by $O(\log \log n)$ then the read/write complexity of these network codes is only a polylogarithmic factor slower than BitTorrent. However, it is better than Network Coding, if the number of rounds is at most $o((\log n - \log p)/\log \delta)$ for p peers, n blocks and in-degree δ.

Optimal Performance.

Table 1 summarizes the results for BitTorrent (BT), Tree Coding (TC) and Network Coding (NC). Note that the maximum average read/write complexity is in any case $O(n^2)$ since they constitute special cases of Network Coding.

The contribution in this paper is two-fold. First we have shown that Tree Coding is in fact efficient for a small number of trees using a divide-and-conquer strategy. Then, we have shown that a clever use of dynamic re-encoding allows optimal performance like full Network Coding. These observations hold for an online model where only the current round is known to the peers. Furthermore, only local knowledge is necessary for the strategies, i.e. the knowledge of data available at the sender and receiver side. For this we have introduced a dynamic version of Tree Coding and have restricted ourselves to use only complete levels of the trees, which allows a perfect balance.

Lower bounds.

We have also shown that BitTorrent fails for some communication graphs of depth 2 even in the offline model where all future communication is known.

Outlook.

The main open question is whether there is an efficient network coding with optimal performance for all communication graphs. If it does not exist one might hope for a network coding with nearly optimal performance and good efficiency, since users would rather compromise on the data distribution than the performance.

The knowledge of the number of transmitted blocks during a round is not very realistic. BitTorrent and Network Coding already work in a stronger online setting where the bandwidth of the current round is not known. It is an open question whether efficient network coding schemes exist in this strong online model.

6. REFERENCES

[1] R. Ahlswede, Ning Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2000.

[2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth content distribution in cooperative environments. In *International Workshop on Peer-to-Peer Systems (IPTPS), LNCS*, volume 2, 2003.

[3] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8):1489–1499, 2002.

[4] P. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proceedings of the 41st Allerton Conference on Communication, Control, and Computing*, September 2003.

[5] Bram Cohen. Incentives build robustness in BitTorrent. Technical report, bittorrent.org, 2003.

[6] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, and C. Liu. Protocol independent Multicast-Sparse Mode (PIM-SM): protocol specification. RFC 2362, Internet Engineering Task Force, June 1998.

[7] Christos Gkantsidis and Pablo Rodriguez. Network coding for large scale content distribution. In *INFOCOM*, pages 2235–2245. IEEE, 2005.

[8] Kirsten Hildrum, John D. Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed object location in a dynamic network. In *SPAA '02: Proceedings of the fourteenth annual ACM symposium on parallel algorithms and architectures*, pages 41–52, New York, NY, USA, 2002. ACM Press.

[9] Dave Levin, Katrina Lacurts, Neil Spring, and Bobby Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. *SIGCOMM Comput. Commun. Rev.*, 38(4):243–254, 2008.

[10] Christian Ortolf, Christian Schindelhauer, and Arne Vater. Classifying peer-to-peer network coding schemes. In *SPAA '09: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 310–318, New York, NY, USA, 2009. ACM.

[11] Christian Ortolf, Christian Schindelhauer, and Arne Vater. Paircoding: Improving File Sharing Using Sparse Network Codes. In *ICIW '09: Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services*, pages 49–57, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in BitTorrent? In *NSDI'07*, Cambridge, MA, April 2007.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Computer Communication Review*, volume 31, pages 161–172. Dept. of Elec. Eng. and Comp. Sci., University of California, Berkeley, 2001.

[14] Sylvia Ratnasamy, Mark Handley, Richard M. Karp, and Scott Shenker. Application-level multicast using content-addressable networks. In Jon Crowcroft and Markus Hofmann, editors, *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*, volume 2233 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2001.

[15] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science, In Proc. of the International Conference on Distributed Systems Platforms (IFIP/ACM),*, 2218:329–350, 2001.

[16] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In Roch Guerin, editor, *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, volume 31, 4 of *Computer Communication Review*, pages 149–160, New York, August 27–31 2001. ACM Press.

[17] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol. RFC 1075, Internet Engineering Task Force, November 1988.

[18] Mea Wang and Baochun Li. How practical is network coding? *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pages 274–278, June 2006.

[19] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV*, June 2001.