使用再造編碼及快取技術之低成本高可用性點對點儲存方法

Regenerating Code Based P2P Storage Scheme with Caching

黎光明，張智維
資訊科學與工程研究所
交通大學
新竹，台灣
kdplant205@gmail.com
jason110570@gmail.com

陳育良，袁賢銘
資訊科學與工程研究所
交通大學
新竹，台灣
imklchen@gmail.com
smyuan@cis.nctu.edu.tw

***Abstract*—High data availability is an important feature of P2P storage system. Erasure coding is a common method employed to support the property. But the technique requires an original file to produce redundant data. On the contrary Regenerating Code, without maintaining the original file, solves the problem through collecting encoded information in a distributed way. However, there is a drawback in this method; that is, encoding needs more peers to get involved in order to obtain good information, which is a harsh condition in a P2P environment due to the requirement of keeping more peers alive simultaneously to hold the file blocks.**

On the condition that the system did not need to store a replica, we recorded the information of peers which have accessed files recently and utilized the data which have been accessed last in the LRU cache of the peer to improve the access performance and reduce the encoding cost. Through simulation, we experimented with different cache sizes under various peer availability. The results showed that our scheme can successfully improve at least 83 percent of the access performance and lower the cost of Regenerating Code when a file is divided into 7 blocks and the cache size is 64 file blocks.

Keywords- p2p, erasure coding, regenerating code, LRU cache.

***摘要*—高資料可靠度是分散式(Peer-to-Peer)儲存系統的重要特性之一。錯誤更正碼(erasure code)是一個常用的方法來產生冗餘(redundancy)的資料，不過這個方法需要原始檔案來產生冗餘的資料。而再造編碼(Regenerating Code)分散式地蒐集多個節點(peer)的資料來產生冗餘，則不需要維護一份原始的檔案。不過這樣做也有一些缺點，存取時無法只透過一個節點(peer)來取得資料，加上編碼時需要更多的節點(peer)同時提供資訊才能達到比傳統方法更好的效率，由於要求同時保持多個活著的節點(peer)去保持檔案區塊，而這在點對點(Peer-to-Peer)的環境下是一個嚴苛的條件。**

系統在不額外維護一份原始檔案的條件下，我們記錄最近存取檔案的節點(peer)並且利用節點中最近最少使用演算法的快取(LRU cache)中的資料來提高存取的效率以及降低編碼的成本。透過用模擬的方式，我們嘗試不同快取大小在不同節點可靠度(peer availability)之下，記錄最近存取檔案的節點的資訊。在實驗中，我們將一個檔案被分成 7 個區塊(block)，並且發現這些被記錄的節點在快取大小為 64 個檔案區塊(block)的條件下，有 83%以上的比例可以改善存取效率和編碼成本。

關鍵字－點對點, 錯誤更正碼, 再造編碼, 最近最少使用快取.

I. 緒論

有許多分散式(Peer-to-Peer)儲存系統的設計，如 OceanStore[1], CFS[2], PAST[3] 以及

Total Recall[4]。對分散式(Peer-to-Peer)儲存系統來說，因為要求提供優質服務，資料可靠度是主要的考量。為了實現高可靠度，副本(replication)方案和錯誤更正碼(erasure code)是常用的方法來產生冗餘(redundancy)的資料以增加可靠度。此外，新的網路編碼應用，再造編碼(Regenerating Code) [5]。不像錯誤更正碼(erasure code)在產生編碼區塊(encoded blocks)前，需要保持完整的檔案。再造編碼(Regenerating Code)透過收集足夠的現有編碼區塊(encoded blocks)產生新的編碼區塊(encoded blocks)。雖然再造編碼(Regenerating Code)比錯誤更正碼(erasure code)有儲存和頻寬的低成本，但也承受較差的存取效能，這是我們要去改善的工作。

下面，我們簡要討論所做的工作在這方面，隨後簡要說明我們如何解決這個問題。然後，在第 4 節，我們將展示實驗和測量結果。

II. 相關研究

關於 P2P(Peer-to-Peer)網路，大多數計算機用戶可能會聯想起一些有名的應用程序，如 BitTorrent 和 Skype。P2P(Peer-to-Peer)網路連接的所有參與者，共享資訊，並利用它們之間的頻寬。有很多 P2P 網路的分類。例如，它可以根據他們的用途分類，如檔案共享，電話，媒體串流和論壇。其他分類，可根據集中化的程度來做分類。在這裡，我們介紹另一個分類，非結構化和結構化 P2P 網路[6]。P2P 網路由所有參與節點(peer)如同網路節點所組成 。如果一個節點(peer)知道另一個節點(peer)的位置，然後利用有方向性的邊(directed edge)將這兩個節點(peer)鏈接起來。這些鏈接連接各個節點形成一個覆蓋網路(overlay network)，如果連接是被固定就成為一個結構化 P2P 網路(structured peer-to-peer network)。他們通常利用 distributed hash table-based (DHT)的索引，如在 Chord system [7]。相反地，非結構化 P2P(Peer-to-Peer)網路，不提供任何對於組織網路的演算法或網路連接的最佳化。在非結構化 P2P 網路，查詢(queries)可能不總是能被解決，因為他們必須以泛流(flooding)方式透過網路去盡可能地找出許多節點(peer)，然而結構化 P2P 網路是經由 distributed hash table-based (DHT)的方式解決查詢(queries)。

在一篇開創性論文[8]中，引起對網路編碼應用的廣泛研究。研究[9] [10]證明線性網路編碼，可以實現最大群組廣播(multicast)能力。在[11] [12]，研究證明，當伽羅瓦代數體(Galois field)的大小足夠大，隨機線性網路編碼可以實現最大組播(multicast)能力。此外，[13]在每邊是單位傳輸量的不循環有向圖上給線性編碼一個確定性多項式時間演算法(deterministic polynomial time algorithms)。 網路編碼應用在單一原始 P2P 檔案的共享在[14]有被提到。網路編碼的關鍵概念是允許在中間網路節點混合資料，以消除資料流的衝突和在中間節點的資料融合過程推翻以往傳統路由選擇過程的概念。網路編碼已經應用到許多領域，如流量，無線資源，安全性，複雜性和抵禦鏈路故障(resilience to link failures)[15]。此外，一些新興的網路編碼應用包括網路監控、開關操作，晶片內建通訊和分散式儲存被概述在[16]。

在[5]表明，再造編碼(Regenerating Code)可以大大減少修補頻寬。它允許一個新的節點下載從倖存的節點中已儲存的資料的功能。圖. 1 顯示一個再造編碼(Regenerating Code)的例子從 [5] 。它是一個 (4,2)-Minimum-Storage Regenerating Code 的例子，4 意味著在系統中每個檔案被分為 4 個編碼區塊以及 2 意味著 4 個編碼區塊中任意 2 個區塊可以重建檔案。線性操作的係數是隨機選取的。在此情況下，修復頻寬是小於原始檔案的大小。相反地，錯誤更正碼(erasure code)必須存取原始檔案來產生一個新的編碼區塊。修補一個編碼區塊的通訊數量，代表標誌為 d，在例子中是等於 3，但是在(4,2)-erasure code 中是等於 2。顯然，d 在再造編碼(Regenerating Code)是大於錯誤更正碼(erasure code)。對於再造編碼(Regenerating Code)，較大的 d 將有利於儲存和減少頻寬的使用[5]。

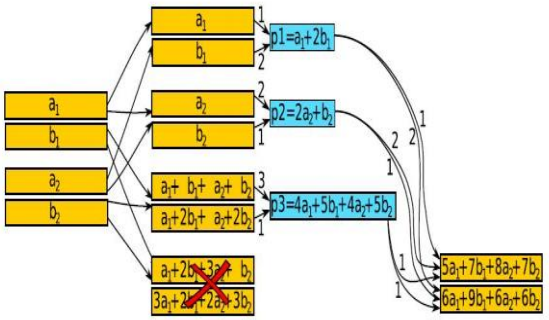

圖 1. A repair for a (4,2)-Minimum-Storage Regenerating Code, from [5]

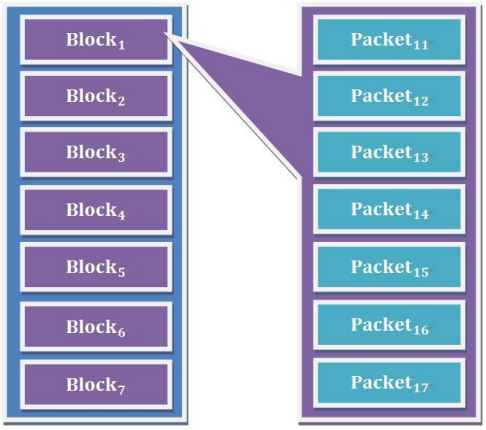

圖 2. The relationship between the blocks and packets in a file

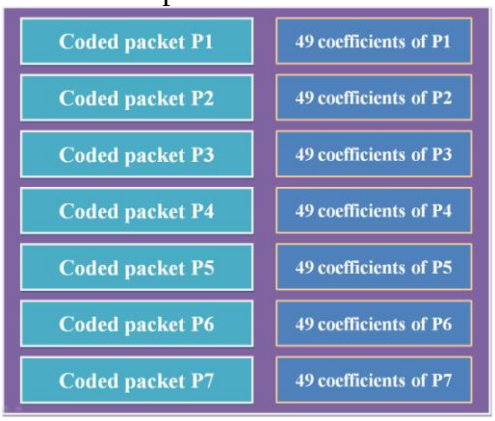

圖 3. The composition of an encoded block

III. 系統設計

在我們的系統設計中，我們假設在 P2P 儲存系統中所有的節點(peers)有兩種類型的儲存空間，永久空間名為“database”和暫存空間名為“LRU cache”。在 database 中的檔案將存在在系統中，直到節點(peer)崩潰或離開，但在 LRU cache 中的檔案，如果空間已滿，這些檔案可能會被替換。替換的規則如同 LRU cache 的演算法。我們的工作採用 Chord [17]以維護路由表(routing tables)的尋找和對每個節點(peer)的查詢管理並且每個檔案有一個唯一的 id。目標檔案被設定為終端用戶所期待的 99. 9%。我們利用(n,7)-Regenerating Code 對檔案進行編碼，其中 n 是根據節點(peer)的可靠度來決定，7 被用於 CFS[2]而且設定通訊數量 d 為 13。如果我們設定 n 為 7，然後每個檔案再被分為 7 個區塊，每個區塊將包含 7 個封包(見圖. 2)

顯然，在一個檔案中總封包數為 49，而且這些封包被用於產生編碼區塊。在編碼區塊中的每個封包也還保存其原始檔案相關的 49 個封包的組合係數(見圖. 3)。對於線性組合，我們選擇一個大的伽羅瓦代數體(Galois field)，以便能成功的解碼，並假設封包大小，也大到足以忽略係數的額外耗損。

為了實現該系統，我們選擇 Minimum-Storage Regenerating (MSR) Codes，它是 MSR 的一個特別案件。從上面的描述，我們得到一對區塊的大小和修補頻寬 $\left(\frac{M}{7}, \frac{13M}{49}\right)$。這裡的 MSR codes 需要與 13 個不同的節點(peers)溝通以進行編碼出一個新的編碼區塊。如果在系統中沒有足夠的節點(peers)，我們改為與錯誤更正碼(erasure code)一樣只收集 7 個不同的區塊。

我們使用的冗餘(redundancy)方案是基於[18]，並使用再造編碼(Regenerating Code)，再加上 LRU cache 和一些 DHT 的資訊。總體而言，每個節點(peer)負責三項工作：索引，註冊和維護。關於維護方面是被底層的 Chord protocol 所支持，在 P2P 儲存系統中，每個節點(peer)控制資料放置，資料查詢和資料可靠

度。對於索引和註冊，每個節點(peer)在其 database 中有一些索引檔案的編碼區塊要定期註冊獨特的檔案 IDs。在這裡，編碼區塊若屬於相同檔案將有相同的 ID，但具有不同係數。當索引收到一份報告，他們在自己的索引表中記錄新的索引上，包括報告節點的 IP 和其檔案的 ID 並設定一個計時器倒數計時，以便移除這些索引。事實上，索引可以利用計時器來檢測被索引的節點(peers)的狀態，換句話說，可以判斷節點(peers)是否活著。第一個索引可以被 DHT(distributed hash table-based)的 hash function 所決定，H(ID)和第一個索引的鄰近索引及後繼索引，如圖. 4 所示。

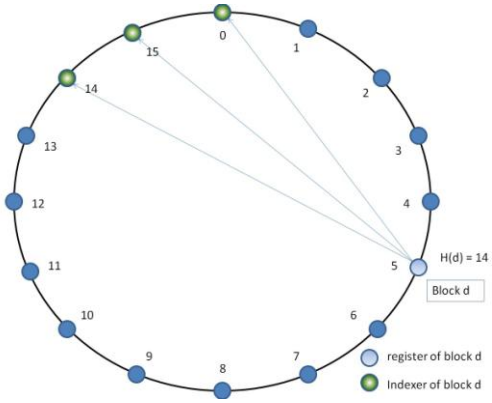

圖 4. A peer registers a block d with 3 indexers

對於維修而言，每個節點(peer)根據 hash function H(ID)，被指派儲存一些檔案的區塊。然後它定期評估在它的索引表中有多少註冊的區塊。如果數量在冗餘(redundancy)門檻之下，它會在與其他索引溝通之前產生新的編碼區塊以增加檔案的可靠度並且報告缺乏冗餘(redundancy)。

接下來，我們要解釋要求一個檔案和產生一個新的編碼區塊的運作。無論哪種運作，每個節點(peer)在從其他節點(peers)請求資訊之前都將檢查其 database 和 LRU cache，並且每個被請求的節點(peer)將透過其本身的 database 和 LRU cache 回應請求檔案的資訊。當一個節點(peer)想要從目標節點(target peer)請求一個檔案時，兩節點的溝通包括兩個步驟。首先，一個節點(peer)節點(target peer)將會和檔案 ID 比較其所有區塊的 ID 以產生區塊資訊的清單，其中包含每個區塊的 49 個係數。其次，請求的節點(peer)就利用此清單從目標節點(target peer)獲得區塊。因此，請求的節點(peer)就可以避免下載到它已經有的區塊。對於請求的運作，索引(indexer)將始終索引最近已經請求檔案的節點(peer)，並傳送節點(peer)和已註冊節點(peer)的資訊給提出請求的節點(peer)。當請求一個檔案時，節點(peer)將選擇第一個索引去獲得擁有檔案的編碼區塊的節點(peer)清單 。在得到節點(peer)清單後，請求節點(peer)根據此清單收集 7 個獨立的區塊。如果資訊還不夠的話，節點將嘗試尋找其他的索引(indexers)。如果所有的索引(indexers)不能提供足夠的資訊，將稍後再試，直到計時器倒數計時到期。

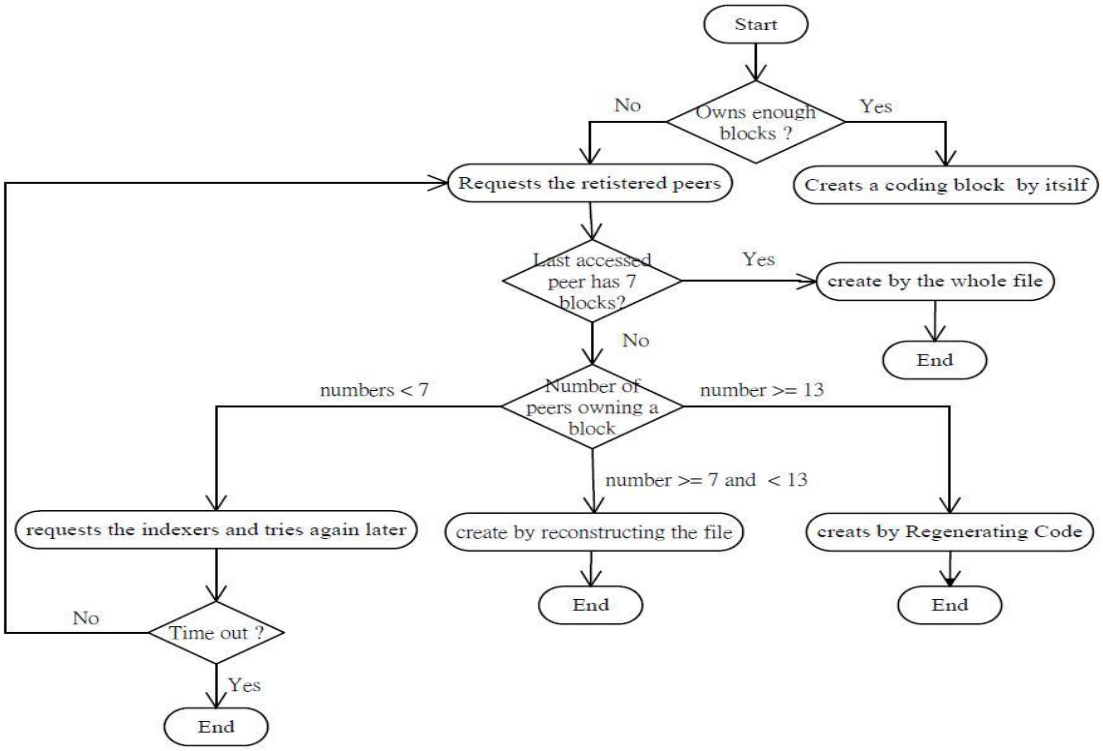

圖 5. The flow char of creating an encoded block

在成功地收集到 7 個獨立的區塊，節點解碼這些區塊並將它們帶回 LRU cache 方便於下次的存取。對於產生一個新的編碼區塊之運作，它類似請求並且如圖. 5 所示。但是，索引(indexers)將索引最近已請求檔案的非註冊節點(peers)而不是索引註冊節點(peers)。

IV. 實驗結果

為了實現我們的計畫，我們利用 P2Psim 軟體，它是一個 discrete event packet level 模擬器可以模擬結構性覆蓋網路(structured overlay networks)。模擬的網路由 1024 個節點(peers)組成，可能交替地離開或加入網路。對每個節點(peer)來說，在連續事件之間的時間間隔，是隨著已知時間的平均指數地分布。在實驗中，每個節點(peer)都有一個 database 和 LRU cache 以及一個索引表(index table)。當一個節點(peer)崩潰或離開，所有的儲存資料和索引(indexers)都將被清除。相反地，如果一個節點(peer)重新加入，它將改變為不同的 IP 和使用不同的 DHT id。在系統中，有 1000 個相同大小的不同檔案，而且所有的檔案具有相同存取的可能性。一開始，所有的檔案隨機分配給這些節點(peers)；而這些節點(peers)將利用檔案產生足夠的編碼區塊，分配給隨機選取的節點(peers)。這裡有兩個重要的觀測值，分別為節點可靠度(peer availability)和快取大小(cache size)，如表. 1 和表. 2 顯示；而且檔案的可靠度被設定為 99.9%。

表 I. INDIVIDUAL PARAMETERS FOR EACH PEER AVAILABILITY

Parameter	Value		
Peer availability	90%	65%	40%
Time (hours)	6	$\frac{90}{65}*6$	$\frac{9}{4}*6$
Data redundancy	2	3	6
Data redundancy (blocks)	14	21	42

表 II. COMMON PARAMETERS FOR ALL PEER AVAILABILITY

Parameter	Value
Cache size (blocks)	8, 16, 32, 64, 128
Average requested blocks per peer	280
Target availability	99.9%

在不影響實驗結果之下，我們先選擇節點(peer)可靠度，然後再設定不同的快取大小(cache size)進行實驗。當節點(peer)可靠度為90%時，實驗時間為 6 個小時。為了觀察節點(peer)可靠度為 40%的資料，我們延長實驗時間到 6*9/4 個小時。在不失準確性之下，在實驗的後半段，我們才開始收集資料。

當存取檔案時，節點(peer)會先檢查自己的 LRU cache。如果沒有 7 個區塊，它將請求索引節點(indexed peer)。如果索引節點(indexed peer)本身 LRU cache 有 7 個區塊，請求節點(peer)只需要建立一個連接去獲得所有的區塊，而不需要建立 7 個連接去收集 7 個區塊。表 3，4，5 顯示當索引節點(peers)攜帶不同的編碼區塊，在不同的快取大小上，對不同的節點可靠度存取的效能。其他案例在這三個表當中代表請求節點(peer)在自己的 LRU cache 中有 7 個區塊。

表 III. THE ACCESS PERFORMANCE FOR PEER AVAILABILITY = 0.9

Block number / Cache siz	The indexed peers have 7 blocks	The indexed peers have partial blocks	The indexed peers have no block	Other cases
8	45.76%	28.44%	25.70%	0.10%
16	69.21%	16.29%	14.30%	0.20%
32	86.35%	7.41%	5.85%	0.39%
64	91.51%	4.62%	3.09%	0.77%
128	91.30%	4.55%	2.67%	1.48%

表 IV. THE ACCESS PERFORMANCE FOR PEER AVAILABILITY = 0.65

Block numb / Cache siz	The indexed peers have 7 blocks	The indexed peers have partial blocks	The indexed peers have no block	Other cases
8	38.72%	30.34%	30.83%	0.10%
16	60.69%	19.89%	19.21%	0.21%
32	80.87%	10.18%	8.58%	0.37%
64	89.06%	5.98%	4.20%	0.76%
128	89.74%	5.34%	3.41%	1.51%

表 V. THE ACCESS PERFORMANCE FOR PEER AVAILABILITY = 0.4

Block numb / Cache siz	The indexed peers have 7 blocks	The indexed peers have partial blocks	The indexed peers have no block	Other cases
8	28.00%	37.96%	33.92%	0.13%
16	48.54%	26.47%	24.78%	0.21%
32	68.97%	16.64%	14.01%	0.38%
64	83.51%	9.18%	6.51%	0.81%
128	85.25%	8.10%	5.07%	1.58%

在我們的實驗中，利用 LRU cache 以提高存取的效能。正如上面的表所顯示的，快取大小最好的選擇為 64，保證存取的效能超出 83%。

V. 結論

為了提高存取效能並降低維護成本，我們為請求區塊操作(request operation)加入註冊節點的索引，並對產生區塊操作(create operation)加入非註冊節點的索引。雖然非註冊節點(peers)的資訊對維護成本沒什麼貢獻，然而註冊節點(peers)的資訊的確幫助提高存取的效能。實驗結果顯示節點(peer)可靠度和快取大小可以改善請求服務，當快取大小為 64 時，達到的效能不會低於 83%。

VI. FUTURE WORK

我們將探索其他實驗的可能性以降低維護成本，同時，我們將把我們的研究應用在 Multimedia Content Discovery and Delivery (mCDN)，並在其中的 P2P 儲存系統實作我們的設計方案。

致謝

本研究為工研院計畫所支持，計畫代碼為 8352B11200 。

參考文獻

[1] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, HakimWeatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. OceanStore: An architecture for global-scale persistent storage. In Proceedings of ACM ASPLOS. ACM, November 2000.

[2] F. Dabek, M. F. Kaashoek, D. Karger, R.Morris, and I. Stoica. Wide-area cooperative storage with CFS. In Proc. ACM SOSP, 2001.

[3] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale,persistent peer-to-peer storage utility. In Proc. SOSP, 2001.

[4] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker. Total recall: System support for automated availability management. In Proc. NSDI, 2004.

[5] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin O. Wainwright and Kannan Ramchandran “Network coding for distributed storage systems” arXiv:0803.0632v1 [cs.NI] 5 Mar 2008.

[6] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," Communications Surveys & Tutorials, IEEE, vol. 7, pp. 72-93, 2005.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proc. ACM SIGCOMM, 2001.

[8] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," IEEE Trans. Inform. Theory, vol. IT-46, pp. 1204–1216, 2000.

[9] M. Médard, R. Koetter, "An Algebraic Approach to Network Coding", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 11, NO. 5, OCTOBER 2003.

[10] S.-Y. R. Li, R. W. Yeung and N. Cai, "Linear network coding," IEEE Trans. Inform. Theory," IT-49: 371-381, 2003.

[11] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 52, NO. 10, OCTOBER 2006

[12] P. Sander, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in Symposium on Parallel Algorithms and. Architectures (SPAA), (San Diego, CA), pp. 286–294, ACM, June 2003.

[13] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," IEEE Trans. Inform. Theory, vol. IT-51, pp. 1973–1982, 2005.

[14] Min Yang Yuanyuan Yang, Peer-to-peer File Sharing Based on Network Coding. Distributed Computing Systems, 2008. ICDCS '08. The 28th International Conference.

[15] C. Fragouli and E. Soljanin, Network Coding Fundamentals. now Publishers, 2007.

[16] C. Fragouli and E. Soljanin, Network Coding Applications. now Publishers, 2008.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proc. ACM SIGCOMM, 2001.

[18] Fan Wu, Tongqing Qiu, Yuequan Chen and Guihai Chen. Redundancy schemes for high availability in DHTs. In Proceedings of ISPA, 2005. Journal version will appear in Journal of Supercomputing, 2007.