



US009270299B2

(12) **United States Patent**  
**Luby et al.**

(10) **Patent No.:** **US 9,270,299 B2**  
(45) **Date of Patent:** **Feb. 23, 2016**

(54) **ENCODING AND DECODING USING ELASTIC CODES WITH FLEXIBLE SOURCE BLOCK MAPPING**

(75) Inventors: **Michael G. Luby**, Berkeley, CA (US); **Payam Pakzad**, Mountain View, CA (US); **Mohammad Amin Shokrollahi**, Preverenges (CH); **Mark Watson**, San Francisco, CA (US); **Lorenzo Vicisano**, Berkeley, CA (US)

4,589,112 A 5/1986 Karim  
4,901,319 A 2/1990 Ross  
5,136,592 A 8/1992 Weng  
5,153,591 A 10/1992 Clark  
5,331,320 A 7/1994 Cideciyan et al.  
5,371,532 A 12/1994 Gelman et al.  
5,372,532 A 12/1994 Robertson, Jr.  
5,379,297 A 1/1995 Glover et al.  
5,421,031 A 5/1995 De Bey

(Continued)

**FOREIGN PATENT DOCUMENTS**

CN 1338839 A 3/2002  
CN 1425228 A 6/2003

(Continued)

**OTHER PUBLICATIONS**

3GPP TS 26.234 V9.1.0, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (Release 9)", Dec. 2009, 179 pages.

(Continued)

(73) Assignee: **QUALCOMM Incorporated**, San Diego, CA (US)

(\* ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 620 days.

(21) Appl. No.: **13/025,900**

(22) Filed: **Feb. 11, 2011**  
(Under 37 CFR 1.47)

**Prior Publication Data**

US 2012/0210190 A1 Aug. 16, 2012

(51) **Int. Cl.**  
**H03M 13/00** (2006.01)  
**H03M 13/37** (2006.01)  
**H04L 1/00** (2006.01)

(52) **U.S. Cl.**  
CPC ..... **H03M 13/3761** (2013.01); **H04L 1/007** (2013.01); **H04L 1/0042** (2013.01); **H04L 1/0057** (2013.01); **H04L 1/0083** (2013.01); **H04L 1/0086** (2013.01)

(58) **Field of Classification Search**  
None  
See application file for complete search history.

**References Cited**

**U.S. PATENT DOCUMENTS**

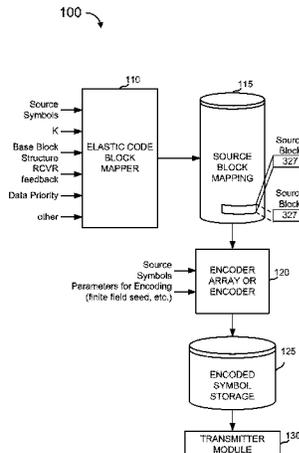
3,909,721 A 9/1975 Bussgang et al.  
4,365,338 A 12/1982 McRae et al.

*Primary Examiner* — Justin R Knapp

(57) **ABSTRACT**

Data can be encoded by assigning source symbols to base blocks, assigning base blocks to source blocks and encoding each source block into encoding symbols, where at least one pair of source blocks is such they have at least one base block in common with both source blocks of the pair and at least one base block not in common with the other source block of the pair. The encoding of a source block can be independent of content of other source blocks. Decoding to recover all of a desired set of the original source symbols can be done from a set of encoding symbols from a plurality of source blocks wherein the amount of encoding symbols from the first source block is less than the amount of source data in the first source block and likewise for the second source block.

**51 Claims, 5 Drawing Sheets**



(56)

## References Cited

## U.S. PATENT DOCUMENTS

5,425,050	A	6/1995	Schreiber et al.	6,411,223	B1	6/2002	Haken et al.
5,432,787	A	7/1995	Chethik	6,415,326	B1	7/2002	Gupta et al.
5,455,823	A	10/1995	Noreen et al.	6,420,982	B1	7/2002	Brown
5,465,318	A	11/1995	Sejnoha	6,421,387	B1	7/2002	Rhee
5,517,508	A	5/1996	Scott	6,430,233	B1	8/2002	Dillon et al.
5,524,025	A	6/1996	Lawrence et al.	6,445,717	B1	9/2002	Gibson et al.
5,566,208	A	10/1996	Balakrishnan	6,459,811	B1	10/2002	Hurst, Jr.
5,568,614	A	10/1996	Mendelson et al.	6,466,698	B1	10/2002	Creusere
5,583,784	A	12/1996	Kapust et al.	6,473,010	B1	10/2002	Vityaev et al.
5,608,738	A	3/1997	Matsushita	6,486,803	B1	11/2002	Luby et al.
5,617,541	A	4/1997	Albanese et al.	6,487,692	B1	11/2002	Morelos-Zaragoza
5,642,365	A	6/1997	Murakami et al.	6,496,980	B1	12/2002	Tillman et al.
5,659,614	A	8/1997	Bailey, III	6,497,479	B1	12/2002	Stoffel et al.
5,699,473	A	12/1997	Kim	6,510,177	B1	1/2003	De Bonet et al.
5,701,582	A	12/1997	DeBey	6,523,147	B1	2/2003	Kroeger et al.
5,751,336	A	5/1998	Aggarwal et al.	6,535,920	B1	3/2003	Parry et al.
5,754,563	A	5/1998	White	6,577,599	B1	6/2003	Gupta et al.
5,757,415	A	5/1998	Asamizuya et al.	6,584,543	B2	6/2003	Williams et al.
5,802,394	A	9/1998	Baird et al.	6,609,223	B1	8/2003	Wolfgang
5,805,825	A	9/1998	Danneels et al.	6,614,366	B2	9/2003	Luby
5,835,165	A	11/1998	Keate et al.	6,618,451	B1	9/2003	Gonikberg
5,844,636	A	12/1998	Joseph et al.	6,631,172	B1	10/2003	Shokrollahi et al.
5,870,412	A	2/1999	Schuster et al.	6,633,856	B2	10/2003	Richardson et al.
5,903,775	A	5/1999	Murray	6,641,366	B2	11/2003	Nordhoff
5,917,852	A	6/1999	Butterfield et al.	6,643,332	B1	11/2003	Morelos-Zaragoza et al.
5,926,205	A	7/1999	Krause et al.	6,677,864	B2	1/2004	Khayrallah
5,933,056	A	8/1999	Rothenberg	6,678,855	B1	1/2004	Gemmell
5,936,659	A	8/1999	Viswanathan et al.	6,694,476	B1	2/2004	Sridharan et al.
5,936,949	A	8/1999	Pasternak et al.	6,704,370	B1	3/2004	Chheda et al.
5,953,537	A	9/1999	Balicki et al.	6,732,325	B1	5/2004	Tash et al.
5,970,098	A	10/1999	Herzberg	6,742,154	B1	5/2004	Barnard
5,983,383	A	11/1999	Wolf	6,748,441	B1	6/2004	Gemmell
5,993,056	A	11/1999	Vaman et al.	6,751,772	B1	6/2004	Kim et al.
6,005,477	A	12/1999	Deck et al.	6,765,866	B1	7/2004	Wyatt
6,011,590	A	1/2000	Saukkonen	6,804,202	B1	10/2004	Hwang
6,012,159	A	1/2000	Fischer et al.	6,810,499	B2	10/2004	Sridharan et al.
6,014,706	A	1/2000	Cannon et al.	6,820,221	B2	11/2004	Fleming
6,018,359	A	1/2000	Kermode et al.	6,831,172	B1	12/2004	Barbucci et al.
6,041,001	A	3/2000	Estakhri	6,849,803	B1	2/2005	Gretz
6,044,485	A	3/2000	Dent et al.	6,850,736	B2	2/2005	McCune, Jr.
6,061,820	A	5/2000	Nakakita et al.	6,856,263	B2	2/2005	Shokrollahi et al.
6,073,250	A	6/2000	Luby et al.	6,868,083	B2	3/2005	Apostolopoulos et al.
6,079,041	A	6/2000	Kunisa et al.	6,876,623	B1	4/2005	Lou et al.
6,079,042	A	6/2000	Vaman et al.	6,882,618	B1	4/2005	Sakoda et al.
6,081,907	A	6/2000	Witty et al.	6,895,547	B2	5/2005	Eleftheriou et al.
6,081,909	A	6/2000	Luby et al.	6,909,383	B2	6/2005	Shokrollahi et al.
6,081,918	A	6/2000	Spielman	6,928,603	B1	8/2005	Castagna et al.
6,088,330	A	7/2000	Bruck et al.	6,937,618	B1	8/2005	Noda et al.
6,097,320	A	8/2000	Kuki et al.	6,956,875	B2	10/2005	Kapadia et al.
6,134,596	A	10/2000	Bolosky et al.	6,965,636	B1	11/2005	DesJardins et al.
6,141,053	A	10/2000	Saukkonen	6,985,459	B2	1/2006	Dickson
6,141,787	A	10/2000	Kunisa et al.	6,995,692	B2	2/2006	Yokota et al.
6,141,788	A	10/2000	Rosenberg et al.	7,010,052	B2	3/2006	Dill et al.
6,154,452	A	11/2000	Marko et al.	7,030,785	B2	4/2006	Shokrollahi et al.
6,163,870	A	12/2000	Luby et al.	7,031,257	B1	4/2006	Lu et al.
6,166,544	A	12/2000	Debbins et al.	7,057,534	B2	6/2006	Luby
6,175,944	B1	1/2001	Urbanke et al.	7,068,681	B2	6/2006	Chang et al.
6,178,536	B1	1/2001	Sorkin	7,068,729	B2	6/2006	Shokrollahi et al.
6,185,265	B1	2/2001	Campanella	7,072,971	B2	7/2006	Lassen et al.
6,195,777	B1	2/2001	Luby et al.	7,073,191	B2	7/2006	Srikantan et al.
6,223,324	B1	4/2001	Sinha et al.	7,100,188	B2	8/2006	Hejna et al.
6,226,259	B1	5/2001	Piret	7,110,412	B2	9/2006	Costa et al.
6,226,301	B1	5/2001	Cheng et al.	7,139,660	B2	11/2006	Sarkar et al.
6,229,824	B1	5/2001	Marko	7,139,960	B2	11/2006	Shokrollahi
6,243,846	B1	6/2001	Schuster et al.	7,143,433	B1	11/2006	Duan et al.
6,272,658	B1	8/2001	Steele et al.	7,151,754	B1	12/2006	Boyce et al.
6,278,716	B1	8/2001	Rubenstein et al.	7,154,951	B2	12/2006	Wang
6,298,462	B1	10/2001	Yi	7,164,370	B1	1/2007	Mishra
6,307,487	B1	10/2001	Luby	7,164,882	B2	1/2007	Poltorak
6,314,289	B1	11/2001	Eberlein et al.	7,168,030	B2	1/2007	Ariyoshi
6,320,520	B1	11/2001	Luby	7,219,289	B2	5/2007	Dickson
6,332,163	B1	12/2001	Bowman-Amuah	7,231,404	B2	6/2007	Paila et al.
6,333,926	B1	12/2001	Van Heeswyk et al.	7,233,264	B2	6/2007	Luby
6,373,406	B2	4/2002	Luby	7,240,236	B2	7/2007	Cutts et al.
6,393,065	B1	5/2002	Piret et al.	7,240,358	B2	7/2007	Horn et al.
				7,243,285	B2	7/2007	Foisy et al.
				7,249,291	B2	7/2007	Rasmussen et al.
				7,254,754	B2	8/2007	Hetzler et al.
				7,257,764	B2	8/2007	Suzuki et al.

(56)		References Cited					
		U.S. PATENT DOCUMENTS					
7,265,688	B2	9/2007	Shokrollahi et al.	2003/0086515	A1	5/2003	Trans et al.
7,293,222	B2	11/2007	Shokrollahi et al.	2003/0101408	A1	5/2003	Martinian et al.
7,295,573	B2	11/2007	Yi et al.	2003/0106014	A1	6/2003	Dohmen et al.
7,304,990	B2*	12/2007	Rajwan ..... 370/389	2003/0138043	A1	7/2003	Hannuksela
7,318,180	B2	1/2008	Starr	2003/0194211	A1	10/2003	Abecassis
7,320,099	B2	1/2008	Miura et al.	2003/0207696	A1	11/2003	Willenegger et al.
7,363,048	B2	4/2008	Cheng et al.	2003/0224773	A1	12/2003	Deeds
7,391,717	B2	6/2008	Klemets et al.	2004/0015768	A1	1/2004	Bordes et al.
7,394,407	B2	7/2008	Shokrollahi et al.	2004/0031054	A1	2/2004	Dankworth et al.
7,398,454	B2	7/2008	Cai et al.	2004/0049793	A1	3/2004	Chou
7,409,626	B1	8/2008	Schelstraete	2004/0081106	A1	4/2004	Bruhn
7,412,641	B2	8/2008	Shokrollahi	2004/0096110	A1	5/2004	Yogeshwar et al.
7,418,651	B2	8/2008	Luby et al.	2004/0117716	A1	6/2004	Shen
7,451,377	B2	11/2008	Shokrollahi	2004/0151109	A1	8/2004	Batra et al.
7,483,447	B2	1/2009	Chang et al.	2004/0162071	A1	8/2004	Grilli et al.
7,483,489	B2	1/2009	Gentric et al.	2004/0207548	A1	10/2004	Kilbank
7,512,697	B2	3/2009	Lassen et al.	2004/0240382	A1	12/2004	Ido et al.
7,525,994	B2	4/2009	Scholte	2004/0255328	A1	12/2004	Baldwin et al.
7,529,806	B1	5/2009	Shteyn	2005/0018635	A1	1/2005	Proctor, Jr.
7,532,132	B2	5/2009	Shokrollahi et al.	2005/0028067	A1	2/2005	Weirauch
7,555,006	B2	6/2009	Wolfe et al.	2005/0071491	A1	3/2005	Seo
7,559,004	B1	7/2009	Chang et al.	2005/0084006	A1	4/2005	Lei et al.
7,570,665	B2	8/2009	Ertel et al.	2005/0091697	A1	4/2005	Tanaka et al.
7,574,706	B2	8/2009	Meulemans et al.	2005/0097213	A1	5/2005	Barrett et al.
7,590,118	B2	9/2009	Giesberts	2005/0105371	A1	5/2005	Johnson et al.
7,597,423	B2	10/2009	Silverbrook	2005/0123058	A1	6/2005	Greenbaum et al.
7,613,183	B1	11/2009	Brewer et al.	2005/0138286	A1	6/2005	Franklin et al.
7,633,413	B2	12/2009	Shokrollahi et al.	2005/0160272	A1	7/2005	Teppler
7,633,970	B2	12/2009	Van Kampen et al.	2005/0163468	A1	7/2005	Takahashi et al.
7,644,335	B2	1/2010	Luby et al.	2005/0180415	A1	8/2005	Cheung et al.
7,650,036	B2	1/2010	Lei et al.	2005/0193309	A1	9/2005	Grilli et al.
7,668,198	B2	2/2010	Yi et al.	2005/0195752	A1	9/2005	Amin et al.
7,676,735	B2	3/2010	Luby et al.	2005/0207392	A1	9/2005	Sivalingham et al.
7,711,068	B2	5/2010	Shokrollahi et al.	2005/0216472	A1	9/2005	Leon et al.
7,720,096	B2	5/2010	Klemets	2005/0216951	A1	9/2005	MacInnis
7,720,174	B2	5/2010	Shokrollahi et al.	2005/0254575	A1	11/2005	Hannuksela et al.
7,721,184	B2	5/2010	Luby et al.	2006/0015568	A1	1/2006	Walsh et al.
7,812,743	B2	10/2010	Luby	2006/0020796	A1	1/2006	Aura et al.
7,831,896	B2	11/2010	Amram et al.	2006/0031738	A1	2/2006	Fay et al.
7,924,913	B2	4/2011	Sullivan et al.	2006/0037057	A1	2/2006	Xu
7,956,772	B2	6/2011	Shokrollahi et al.	2006/0093634	A1	5/2006	Lutz et al.
7,961,700	B2	6/2011	Malladi et al.	2006/0107174	A1	5/2006	Heise
7,971,129	B2	6/2011	Watson et al.	2006/0109805	A1	5/2006	Malamal Vadakital et al.
7,979,769	B2	7/2011	Lee et al.	2006/0120464	A1	6/2006	Hannuksela
8,027,328	B2	9/2011	Yang et al.	2006/0212444	A1	9/2006	Handman et al.
8,028,322	B2	9/2011	Riedl et al.	2006/0212782	A1	9/2006	Li
8,081,716	B2	12/2011	Kang et al.	2006/0229075	A1	10/2006	Kim et al.
8,135,073	B2	3/2012	Shen	2006/0244824	A1	11/2006	Debey
8,185,794	B2	5/2012	Lohmar et al.	2006/0244865	A1	11/2006	Simon
8,185,809	B2	5/2012	Luby et al.	2006/0248195	A1	11/2006	Toumura et al.
RE43,741	E	10/2012	Shokrollahi et al.	2006/0256851	A1	11/2006	Wang et al.
8,301,725	B2	10/2012	Biderman et al.	2007/0002953	A1	1/2007	Kusunoki
8,327,403	B1	12/2012	Chilvers et al.	2007/0006274	A1	1/2007	Paila et al.
8,340,133	B2	12/2012	Kim et al.	2007/0016594	A1	1/2007	Visharam et al.
8,422,474	B2	4/2013	Park et al.	2007/0022215	A1	1/2007	Singer et al.
8,462,643	B2	6/2013	Walton et al.	2007/0028099	A1	2/2007	Entin et al.
8,544,043	B2	9/2013	Parekh et al.	2007/0078876	A1	4/2007	Hayashi et al.
8,572,646	B2	10/2013	Haberman et al.	2007/0081562	A1	4/2007	Ma
8,615,023	B2	12/2013	Oh et al.	2007/0110074	A1	5/2007	Bradley et al.
8,638,796	B2	1/2014	Dan et al.	2007/0140369	A1	6/2007	Limberg et al.
8,713,624	B1	4/2014	Harvey et al.	2007/0162568	A1	7/2007	Gupta et al.
8,737,421	B2	5/2014	Zhang et al.	2007/0162611	A1	7/2007	Yu et al.
8,812,735	B2	8/2014	Igarashi	2007/0176800	A1	8/2007	Rijavec
2001/0015944	A1	8/2001	Takahashi et al.	2007/0177811	A1	8/2007	Yang et al.
2001/0033586	A1	10/2001	Takashimizu et al.	2007/0185973	A1	8/2007	Wayda et al.
2002/0009137	A1	1/2002	Nelson et al.	2007/0195894	A1	8/2007	Shokrollahi et al.
2002/0053062	A1	5/2002	Szymanski	2007/0200949	A1	8/2007	Walker et al.
2002/0083345	A1	6/2002	Halliday et al.	2007/0201549	A1	8/2007	Hannuksela et al.
2002/0085013	A1	7/2002	Lippincott	2007/0204196	A1	8/2007	Watson et al.
2002/0133247	A1	9/2002	Smith et al.	2007/0230568	A1	10/2007	Eleftheriadis et al.
2002/0141433	A1	10/2002	Kwon et al.	2007/0233784	A1	10/2007	ORourke et al.
2002/0143953	A1	10/2002	Aiken	2007/0255844	A1	11/2007	Shen et al.
2002/0191116	A1	12/2002	Kessler et al.	2007/0277209	A1	11/2007	Yousef
2003/0005386	A1	1/2003	Bhatt et al.	2008/0010153	A1	1/2008	Pugh-O'Connor et al.
2003/0037299	A1	2/2003	Smith	2008/0034273	A1	2/2008	Luby
				2008/0052753	A1	2/2008	Huang et al.
				2008/0058958	A1	3/2008	Cheng
				2008/0059532	A1	3/2008	Kazmi et al.
				2008/0066136	A1	3/2008	Dorai et al.



(56)

## References Cited

## FOREIGN PATENT DOCUMENTS

JP	2001036417	2/2001	KR	1020030074386	A	9/2003
JP	2001094625	4/2001	KR	20040107152	A	12/2004
JP	2001223655	A 8/2001	KR	20040107401	A	12/2004
JP	2001251287	A 9/2001	KR	20050009376	A	1/2005
JP	2001274776	A 10/2001	KR	100809086	B1	3/2008
JP	2001274855	A 10/2001	KR	20080083299	A	9/2008
JP	2002073625	A 3/2002	KR	20090098919	A	9/2009
JP	2002204219	A 7/2002	RU	99117925	A	7/2001
JP	2002543705	A 12/2002	RU	2189629	C2	9/2002
JP	2003507985	2/2003	RU	2265960	C2	12/2005
JP	2003092564	A 3/2003	RU	2290768	C1	12/2006
JP	2003510734	A 3/2003	RU	2297663	C2	4/2007
JP	2003174489	6/2003	RU	2312390	C2	12/2007
JP	2003256321	A 9/2003	RU	2357279	C2	5/2009
JP	2003318975	A 11/2003	TW	1246841	B	1/2006
JP	2003319012	11/2003	TW	1354908		12/2011
JP	2003333577	A 11/2003	TW	1355168		12/2011
JP	2004048704	A 2/2004	WO	WO9634463	A1	10/1996
JP	2004070712	A 3/2004	WO	WO-9750183	A1	12/1997
JP	2004135013	A 4/2004	WO	WO9804973	A1	2/1998
JP	2004165922	A 6/2004	WO	WO9832231		7/1998
JP	2004516717	A 6/2004	WO	WO-9832256	A1	7/1998
JP	2004192140	A 7/2004	WO	WO0014921	A1	3/2000
JP	2004193992	A 7/2004	WO	WO0018017		3/2000
JP	2004529533	A 9/2004	WO	WO0052600	A1	9/2000
JP	2004289621	A 10/2004	WO	WO0120786	A1	3/2001
JP	2004343701	A 12/2004	WO	WO0157667	A1	8/2001
JP	2004348824	A 12/2004	WO	WO0158130	A2	8/2001
JP	2004362099	A 12/2004	WO	WO0158131	A2	8/2001
JP	2005094140	A 4/2005	WO	WO0227988	A2	4/2002
JP	2005136546	A 5/2005	WO	WO0247391		6/2002
JP	2005514828	T 5/2005	WO	02063461	A1	8/2002
JP	2005204170	A 7/2005	WO	WO-03046742	A1	6/2003
JP	2005223433	A 8/2005	WO	WO03056703		7/2003
JP	2005277950	A 10/2005	WO	WO03105350		12/2003
JP	2006503463	A 1/2006	WO	WO-03105484	A1	12/2003
JP	2006505177	A 2/2006	WO	WO2004015948	A1	2/2004
JP	2006506926	A 2/2006	WO	WO2004019521	A1	3/2004
JP	2006074335	A 3/2006	WO	WO2004030273	A1	4/2004
JP	2006074421	A 3/2006	WO	WO2004034589	A2	4/2004
JP	2006115104	A 4/2006	WO	WO-2004036824	A1	4/2004
JP	3809957	6/2006	WO	WO2004040831	A1	5/2004
JP	2006174032	A 6/2006	WO	WO-2004047019	A2	6/2004
JP	2006174045	A 6/2006	WO	WO2004047455	A1	6/2004
JP	2006186419	A 7/2006	WO	WO-2004088988	A1	10/2004
JP	2006519517	A 8/2006	WO	WO-2004109538	A1	12/2004
JP	2006287422	A 10/2006	WO	WO2005036753	A2	4/2005
JP	2006319743	A 11/2006	WO	WO2005041421	A1	5/2005
JP	2007013675	A 1/2007	WO	WO2005078982	A1	8/2005
JP	2007089137	A 4/2007	WO	WO-2005107123		11/2005
JP	3976163	6/2007	WO	WO2005112250	A2	11/2005
JP	2007158592	A 6/2007	WO	WO-2006013459	A1	2/2006
JP	2007174170	A 7/2007	WO	WO2006020826	A2	2/2006
JP	2007520961	A 7/2007	WO	WO-2006036276		4/2006
JP	2007228205	A 9/2007	WO	2006060036	A1	6/2006
JP	2008011404	A 1/2008	WO	WO-2006057938	A2	6/2006
JP	2008016907	A 1/2008	WO	WO2006084503	A1	8/2006
JP	2008508761	A 3/2008	WO	WO-2006116102	A2	11/2006
JP	2008508762	A 3/2008	WO	WO-2006135878	A2	12/2006
JP	2008283232	A 11/2008	WO	2007078253	A2	7/2007
JP	2008283571	A 11/2008	WO	WO2007090834	A2	8/2007
JP	2008543142	A 11/2008	WO	WO-2007098397	A2	8/2007
JP	2008546361	A 12/2008	WO	WO-2007098480	A1	8/2007
JP	2009027598	A 2/2009	WO	2008011549	A2	1/2008
JP	2009522921	A 6/2009	WO	WO-2008023328	A3	4/2008
JP	2009522922	A 6/2009	WO	WO2008054100	A1	5/2008
JP	2009171558	A 7/2009	WO	2008086313	A1	7/2008
JP	2009527949	A 7/2009	WO	WO2008085013	A1	7/2008
JP	2009277182	A 11/2009	WO	WO-2008131023	A1	10/2008
JP	2009544991	A 12/2009	WO	2008144004	A1	11/2008
JP	2010539832	A 12/2010	WO	WO-2009065526	A1	5/2009
JP	2008502212	A 1/2011	WO	WO-2009137705	A2	11/2009
JP	2001189665	A 2/2011	WO	2009143741	A1	12/2009
JP	2011087103	A 4/2011	WO	WO2010085361	A2	7/2010
KR	1020030071815	9/2003	WO	WO-2011038013		3/2011
			WO	WO-2011038034	A1	3/2011
			WO	2011059286	A2	5/2011

(56)

## References Cited

## FOREIGN PATENT DOCUMENTS

WO	2011070552 A1	6/2011
WO	2011102792 A1	8/2011
WO	WO-2012021540	2/2012

## OTHER PUBLICATIONS

3GPP TS 26.244 V9.1.0, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP), (Release 9), Mar. 2010, 55 pages.

3GPP TS 26.247, v1.5.0, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) (Release 10), 2010, 91 pages.

3rd Generation Partnership Project, Technical Specification Group Services and System Aspects Transparent end-to-end packet switched streaming service (PSS), 3GPP file format (3GP) (Release 9), 3GPP Standard, 3GPP TS 26.244, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre, 650, Route Des Lucioles, F-06921 Sophia-Antipolis Cedex, France, No. V8.1.0, Jun. 1, 2009, pp. 1-52, XP050370199.

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP) (Release 9), 3GPP Standard; 3GPP TS 26.244, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre; 650, Route Des Lucioles; F-06921 Sophia-Antipolis Cedex; France, No. V9.2.0, Jun. 9, 2010, pp. 1-55, XP050441544, [retrieved on Jun. 9, 2010].

Aggarwal, C. et al.: "A Permutation-Based Pyramid Broadcasting Scheme for Video-on-Demand Systems," Proc. IEEE Int'l Conf. on Multimedia Systems, Hiroshima, Japan (Jun. 1996).

Aggarwal, C. et al.: "On Optimal Batching Policies for Video-on-Demand Storage Servers," Multimedia Systems, vol. 4, No. 4, pp. 253-258 (1996).

Albanese, A., et al., "Priority Encoding Transmission", IEEE Transactions on Information Theory, vol. 42, No. 6, pp. 1-22, (Nov. 1996).  
Alex Zambelli, "IIS Smooth Streaming Technical Overview", Microsoft Mar. 25, 2009, XP002620446, Retrieved from the Internet: URL:<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=03d22583-3ed6-44da-8464-b1b4b5ca7520>, [retrieved on Jan. 21, 2011].

Almeroth, et al., "The use of multicast delivery to provide a scalable and interactive video-on-demand service", IEEE Journal on Selected Areas in Communication, 14(6): 1110-1122, (1996).

Alon, et al.: "Linear Time Erasure Codes with Nearly Optimal Recovery," Proceedings of the Annual Symposium on Foundations of Computer Science, US, Los Alamitos, IEEE Comp. Soc. Press, vol. Symp. 36, pp. 512-516 (Oct. 23, 1995) XP000557871.

Amin Shokrollahi: "LDPC Codes: An Introduction" Internet Citation Apr. 2, 2003, XP002360065 Retrieved from the Internet: URL: [http://www.ipm.ac.ir/IPM/homepage/Amin\\_2.pdf](http://www.ipm.ac.ir/IPM/homepage/Amin_2.pdf) [retrieved on Dec. 19, 2005].

Amon P et al., "File Format for Scalable Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, IEEE Service Center, Piscataway, NJ, US, vol. 17, No. 9, Sep. 1, 2007, pp. 1174-1185, XP011193013, ISSN: 1051-8215, DOI:10.1109/TCSVT.2007.905521.

Anonymous: [Gruneberg, K., Narasimhan, S. and Chen, Y., editors] "Text of ISO/IEC 13818-1:2007/PDAM 6 MVC operation point descriptor", 90 MPEG Meeting; Oct. 26, 2009-Oct. 30, 2009; Xian; (Motion Picture Expertgroup or ISO/IEC JTC1/SC29/WG11), No. N10942, Nov. 19, 2009, XP030017441.

Anonymous: "Text of ISO/IEC 14496-12 3rd Edition", 83 MPEG Meeting; Jan. 14, 2008-Jan. 18, 2008; Antalya; (Motion Pictureexpert Group or ISO/IEC JTC1/SC29/WG11), No. N9678, Apr. 22, 2008, XP030016172.

Anonymous: "Text of ISO/IEC 14496-15 2nd edition", 91 MPEG Meeting; Jan. 18, 2010-Jan. 22, 2010; Kyoto; (Motion Picture Expertgroup or ISO/IEC JTC1/SC29/WG11) No. N11139, Jan. 22, 2010, XP030017636.

Bar-Noy, et al., "Competitive on-line stream merging algorithms for media-on-demand", Draft (Jul. 2000), pp. 1-34.

Bar-Noy et al. "Efficient algorithms for optimal stream merging for media-on-demand," Draft (Aug. 2000), pp. 1-43.

Bigloo, A. et al.: "A Robust Rate-Adaptive Hybrid ARQ Scheme and Frequency Hopping for Multiple-Access Communication Systems," IEEE Journal on Selected Areas in Communications, US, IEEE Inc, New York (Jun. 1, 1994) pp. 889-893, XP000464977.

Bitner, J.R., et al.: "Efficient Generation of the Binary Reflected Gray code and Its Applications," Communications of the ACM, pp. 517-521, vol. 19 (9), 1976.

Blomer, et al., "An XOR-Based Erasure-Resilient Coding Scheme," ICSI Technical Report No. TR-95-048 (1995) [avail. at <ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-048.pdf>].

Bross, et al., "High efficiency video coding (HEVC) text specification draft 6," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCTVC-I1003, 7th Meeting: Geneva, CH, Nov. 21-30, 2011, pp. 259.

Bross, et al., "High efficiency video coding (HEVC) text specification draft 7," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 9th Meeting: Geneva, CH, Apr. 27-May 7, 2012, JCTVC-I1003\_d21, pp. 290.

Bross, et al., "High efficiency video coding (HEVC) text specification draft 8," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 10th Meeting: Stockholm, SE, Jul. 11-20, 2012, JCTVC-I1003\_d7, pp. 261.

Bross et al., "WD4: Working Draft 4 of High-Efficiency Video Coding," JCTVC-F803\_d2, (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 Joint Collaborative Team on Video Coding, 6th Meeting, Torino, IT, Jul. 14-22, 2011, 226 pages.

Bross et al., "WD5: Working Draft 5 of High-Efficiency Video Coding," JCTVC-G1103\_d2, (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 Joint Collaborative Team on Video Coding, 7th Meeting, Geneva, Switzerland (Nov. 2011), 214 pages.

Byers, J.W. et al.: "A Digital Fountain Approach to Reliable Distribution of Bulk Data," Computer Communication Review, Association for Computing Machinery, New York, US, vol. 28, No. 4 (Oct. 1998) pp. 56-67 XP000914424 ISSN:0146-4833.

Byers, J.W. et al.: "Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads," 1999, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 275-283, Mar. 21, 1999, XP000868811.

Cataldi et al., "Sliding-Window Raptor Codes for Efficient Scalable Wireless Video Broadcasting With Unequal Loss Protection", IEEE Transactions on Image Processing, Jun. 1, 2010, pp. 1491-1503, vol. 19, No. 6, IEEE Service Center, XP011328559, ISSN: 1057-7149, DOI: 10.1109/TIP.2010.2042985.

Charles Lee L.H., "Error-Control Block Codes for Communications Engineers", 2000, Artech House, XP002642221 pp. 39-45.

Chen et al., "Response to the CfP on HTTP Streaming: Adaptive Video Streaming based on AVC", 93. MPEG Meeting; Jul. 26, 2010-Jul. 30, 2010; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11), No. M17909, Jul. 26, 2010, XP030046499.

Choi S: "Temporally enhanced erasure codes for reliable communication protocols" Computer Networks, Elsevier Science Publishers B.V., Amsterdam, NL, vol. 38, No. 6, Apr. 22, 2002, pp. 713-730, XP004345778, ISSN: 1389-1286, DOI:10.1016/S1389-1286(01)00280-8.

Clark G.C., et al., "Error Correction Coding for Digital Communications, System Applications," Error Correction Coding for Digital Communications, New York, Plenum Press, US, Jan. 1, 1981, pp. 331-341.

D. Gozalvez et al. "AL-FEC for Improved Mobile Reception of MPEG-2 DVB-Transport Streams" Hindawi Publishing Corporation, International Journal of Digital Multimedia Broadcasting vol. 2009, Dec. 31, 2009, pp. 1-10, XP002582035 Retrieved from the Internet: URL:<http://www.hindawi.com/journals/ijdmb/2009/614178.html> [retrieved on May 12, 2010].

(56)

## References Cited

## OTHER PUBLICATIONS

- Dan, A. et al.: "Scheduling Policies for an On-Demand Video Server with Batching," Proc. ACM Multimedia, pp. 15-23 (Oct. 1998).
- Davey, M.C. et al.: "Low Density Parity Check Codes over GF(q)" IEEE Communications Letters, vol. 2, No. 6 pp. 165-167 (1998).
- Digital Fountain: "Specification Text for Raptor Forward Error Correction," TDOC S4-050249 of 3GPP TSG SA WG 4 Meeting #34 [Online] (Feb. 25, 2005) pp. 1-23, XP002425167, Retrieved from the Internet: URL: [http://www.3gpp.org/ftp/tsg\\_sa/WG4\\_CODECS/TSGS4\\_34/Docs](http://www.3gpp.org/ftp/tsg_sa/WG4_CODECS/TSGS4_34/Docs).
- Digital Fountain: "Raptor code specification for MBMS file download," 3GPP SA4 PSM AD-HOC #31 (May 21, 2004) XP002355055 pp. 1-6.
- "Digital Video Broadcasting (DVB); Guidelines for the implementation of DVB-IP Phase 1 specifications; ETSI TS 102 542" ETSI Standards, LIS, Sophia Antipolis cedex, France, vol. BC, No. V1.2.1, Apr. 1, 2008, XP014041619 ISSN: 0000-0001 p. 43 p. 66 pp. 70, 71. DVB-IP Standard: DVB BlueBook A086r4 (Mar. 2007) Transport of MPEG 2 Transport Stream (TS) Based DVB Services over IP Based Networks, ETSI Technical Specification 102 034 v1.3.1.
- Eager, et al. "Minimizing bandwidth requirements for on-demand data delivery," Proceedings of the International Workshop on Advances in Multimedia Information Systems, p. 80-87 (Indian Wells, CA Oct. 1999).
- Eager, et al., "Optimal and efficient merging schedules for video-on-demand servers," Proc. ACM Multimedia, vol. 7, pp. 199-202 (1999).
- Esaki, et al.: "Reliable IP Multicast Communication Over ATM Networks Using Forward Error Correction Policy," IEICE Transactions on Communications, JP, Institute of Electronics Information and Comm. ENG. Tokyo, vol. E78-V, No. 12, (Dec. 1995), pp. 1622-1637, XP000556183.
- Feng, G., Error Correcting Codes over Z<sub>2</sub><sup>m</sup> for Algorithm-Based Fault-Tolerance, IEEE Transactions on Computers, vol. 43, No. 3, Mar. 1994, pp. 370-374.
- Fernando, et al., "httpstreaming of MPEG Media—Response to CFP", 93 MPEG Meeting; Jul. 26, 2010-Jul. 30, 2010; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SCE29/WG11), No. M17756, Jul. 22, 2010, XP030046346.
- Fielding et al., "RFC 2616: Hypertext Transfer Protocol HTTP/1.1", Internet Citation, Jun. 1999 (Jun. 1999), pp. 165, XP002196143, Retrieved from the Internet: URL: <http://www.rfc-editor.org/> [retrieved on Apr. 15, 2002].
- Gao, L. et al.: "Efficient Schemes for Broadcasting Popular Videos," Proc. Inter. Workshop on Network and Operating System Support for Digital Audio and Video, pp. 1-13 (1998).
- Gasiba, Tiago et al., "System Design and Advanced Receiver Techniques for MBMS Broadcast Services" Proc. 2006 International Conference on Communications (ICC 2006), Jun. 1, 2006, pp. 5444-5450, XP031025781 ISBN: 978-1-4244-0354-7.
- Gemmell, et al., "A Scalable Multicast Architecture for One-To-Many Telepresentations", Multimedia Computing and Systems, 1998/Proceedings. IEEE International Conference on Austin, TX, USA Jun. 28-Jul. 1, 1998, Los Alamitos, CA USA, IEEE Comput. Soc, US, Jun. 28, 1998, pp. 128-139, XP010291559.
- Goyal: "Multiple Description Coding: Compression Meets the Network," In Signal Processing Magazine, IEEE, vol. 18., Issue 5 (Sep. 2001) pp. 74-93 URL: [http://www.rle.mit.edu/stir/documents/Goyal\\_SigProcMag2001\\_MD.pdf](http://www.rle.mit.edu/stir/documents/Goyal_SigProcMag2001_MD.pdf) [Nov. 4, 2007].
- Gozalvez D et, al: "Mobile reception of DVB-T services by means of AL-FEC protection" Proc. IEEE Intern. Symposium on Broadband Multimedia Systems and Broadcasting (BMSB '09), IEEE, Piscataway, NJ, USA, May 13, 2009, pp. 1-5, XP031480155 ISBN: 978-1-4244-2590-7.
- Gracie et al., "Turbo and Turbo-Like Codes: Principles and Applications in Telecommunications", Proceedings of the IEEE, Jun. 1, 2007, pp. 1228-1254, vol. 95, No. 6, IEEE, XP011189323, ISSN: 0018-9219, DOI: 10.1109/JPR0C.2007.895197.
- Hagenauer, J.: "Soft is better than hard" Communications, Coding and Cryptology, Kluwer Publication May 1994, XP002606615
- Retrieved from the Internet : URL: <http://www.int.ei.turn.de/veroeffentlichungen/1994/cc94h.pdf> [retrieved on Oct. 25, 2010].
- Hershey, et al., "Random Parity Coding (RPC)", 1996 IEEE International Conference on Communications (ICC). Converging Technologies for Tomorrow's Applications. Dallas, Jun. 23-27, 1996, IEEE International Conference on Communications (ICC), New York, IEEE, US, vol. 1, Jun. 23, 1996, pp. 122-126, XP000625654.
- Hua, et al., "Skyscraper broadcasting: A new broadcasting system for metropolitan video-on-demand systems", Proc. ACM SIGCOMM, pp. 89-100 (Cannes, France, 1997).
- Huawei et al., "Implicit mapping between CCE and PUCCH for ACK/NACK TDD", 3GPP Draft; R1-082359, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre; 650, Route Des Lucioles ; F-06921 Sophia-Antipolis Cedex ; France, vol. RAN WG1, no. Warsaw, Poland, Jun. 24, 2008, XP050110650, [retrieved on Jun. 24, 2008] .
- IETF RFC 2733: Rosenberg, J. et al. "An RTP Payload Format for Generic Forward Error Correction," Network Working Group, RFC 2733 (Dec. 1999).
- International Search Report and Written Opinion—PCT/US2012/024737—ISA/EPO—May 11, 2012.
- International Search Report and Written Opinion—PCT/US2012/024755—ISA/EPO—Apr. 23, 2012.
- International Standard ISO/IEC 14496-12, Information Technology—Coding of audio-visual objects—Part 12: ISO base media file format, Third Edition, Oct. 15, 2008, 120 pp.
- ISO/IEC JTC 1/SC 29, ISO/IEC FCD 23001-6, Information technology—MPEG systems technologies—Part 6: Dynamic adaptive streaming over HTTP (DASH), Jan. 28, 2011.
- ITU-T H.264, Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services—Coding of moving video, Advanced video coding for generic audiovisual services, The International Telecommunication Union. Jun. 2011, 674 pp.
- Jiang., File Format for Scalable Video Coding, PowerPoint Presentation for CMPT 820, Summer 2008.
- Jin Li, "The Efficient Implementation of Reed-Solomon High Rate Erasure Resilient Codes" Proc. 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia, PA, USA, IEEE, Piscataway, NJ, vol. 3, Mar. 18, 2005, pp. 1097-1100, XP010792442, DOI: 10.1109/ICASSP.2005.1415905 ISBN: 978-0-7803-8874-1.
- Juhn, L. et al.: "Adaptive Fast Data Broadcasting Scheme for Video-on-Demand Service," IEEE Transactions on Broadcasting, vol. 44, No. 2, pp. 182-185 (Jun. 1998).
- Juhn, L. et al.: "Harmonic Broadcasting for Video-on-Demand Service," IEEE Transactions on Broadcasting, vol. 43, No. 3, pp. 268-271 (Sep. 1997).
- Kallel, "Complementary Punctured Convolutional (CPC) Codes and Their Applications", IEEE Transactions on Communications, IEEE Inc., New York, US, Vol. 43, No. 6, Jun. 1, 1995, pp. 2005-2009.
- Kimata H et al., "Inter-View Prediction With Downsampled Reference Pictures", ITU Study Group 16—Video Coding Experts Group—ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q6), No. JVT-W079, Apr. 19, 2007, XP030007039.
- Kimura et al., "A Highly Mobile SDM-OFDM System Using Reduced-Complexity-and-Processing", IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Sep. 1, 2007, pp. 1-5, IEEE, XP031168836, ISBN: 978-1-4244-1143-6, DOI: 10.1109/PIMRC.2007.4394758.
- Kozamernik F: "Media streaming over the Internet", Internet Citation, Oct. 2002, XP002266291, Retrieved from the Internet: URL: [http://www.ebu.ch/trev\\_292-kozamernik.pdf](http://www.ebu.ch/trev_292-kozamernik.pdf) [retrieved on Jan. 8, 2004] section "Video codecs for scalable streaming".
- Lee L., et al., "VLSI implementation for low density parity check decoder", Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems, 2001. ICECS 2001, Sep. 2, 2001, vol. 3, pp. 1223-1226.
- Lin, S. et al.: "Error Control Coding-Fundamentals and Applications," 1983, Englewood Cliffs, pp. 288, XP002305226.
- Luby Digital Fountain A Shokrollahi E Pfl M Watson Digital Fountain T Stockhammer Nomor Research M: "Raptor Forward Error Correc-



(56)

## References Cited

## OTHER PUBLICATIONS

Pyle, et al., "Microsoft http smooth Streaming: Microsoft response to the Call for Proposal on httpstreaming", 93 MPEG Meeting; Jul. 26, 2010-Jul. 30, 2010; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SCE29/WG11), No. M17902, Jul. 22, 2010, XP030046492.

Qualcomm Europe S A R L: "Baseline Architecture and Definitions for HTTP Streaming", 3GPP Draft; S4-090603\_HTTP\_Streaming\_Architecture, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre; 650, Route Des Lucioles; F-06921 Sophia-Antipolis Cedex; France, no. Kista; 20090812, Aug. 12, 2009, XP050356889.

Qualcomm Incorporated: "Use Cases and Examples for Adaptive httpstreaming", 3GPP Draft; S4-100408-USECASES-HSD, 3rd Generation Partnership Project (JGPP), Mobile Competence Centre; 650, Route Des Lucioles; F-06921 Sophia-Antipolis Cedex; France, vol. SA WG4, no. Prague, Czech Republic; 20100621, Jun. 17, 2010, XP050438085, [retrieved on Jun. 17, 2010].

Rangan, et al., "Designing an On-Demand Multimedia Service," IEEE Communication Magazine, vol. 30, pp. 56-64, (Jul. 1992).

Realtworks Inc et al., "Format for HTTP Streaming Media Presentation Description", 3GPP Draft; S4-100020, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre; 650, Route Des Lucioles; F-06921 Sophia-Antipolis Cedex; France, vol. SA WG4, no. St Julians, Malta; 20100125, Jan. 20, 2010, XP050437753, [retrieved on Jan. 20, 2010].

Research in Motion UK Limited: "An MPD delta file for httpstreaming", 3GPP Draft; S4-100453, 3rd Generation Partnership Project (SGPP), Mobile Competence Centre; 650, Route Des Lucioles; F-06921 Sophia-Antipolis Cedex; France, vol. SA WG4, no. Prague, Czech Republic; 20100621, Jun. 16, 2010, XP050438066, [retrieved on Jun. 16, 2010].

Rhyu, et al., "Response to Call for Proposals on httpstreaming of MPEG Media", 93 MPEG Meeting; Jul. 26, 2010-Jul. 30, 2010; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SCE29/WG11) No. M17779, Jul. 26, 2010, XP030046369.

Rizzo, L. "Effective Erasure Codes for Reliable Computer Communication Protocols," Computer Communication Review, 27 (2) pp. 24-36 (Apr. 1, 1997), XP000696916.

Roca, V. et al.: "Design, Evaluation and Comparison of Four Large Block FEC Codecs, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec," INRIA Research Report RR-5225 (2004).

Roca, V., et al. "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes", IETF RFC 5170 (Jun. 2008), pp. 1-34.

Rost, S. et al., "The Cyclone Server Architecture: streamlining delivery of popular content," 2002, Computer Communications, vol. 25, No. 4, pp. 1-10.

Roth, R., et al., "A Construction of Non-Reed-Solomon Type MDS Codes", IEEE Transactions of Information Theory, vol. 35, No. 3, May 1989, pp. 655-657.

Roth, R., "On MDS Codes via Cauchy Matrices", IEEE Transactions on Information Theory, vol. 35, No. 6, Nov. 1989, pp. 1314-1319.

Seshan, S. et al., "Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience," Wireless Personal Communications, NL; Kluwer Academic Publishers, vol. 4, No. 2 (Mar. 1, 1997) pp. 141-162, XP000728589.

Shacham: "Packet Recovery and Error Correction in High-Speed Wide-Area Networks," Proceedings of the Military Communications Conference. (Milcom), US, New York, IEEE, vol. 1, pp. 551-557 (1989) XP000131876.

Shierl T; Gruneberg K; Narasimhan S; Vetro A: "ISO/IEC 13818-1:2007/FPDAM 4—Information Technology Generic Coding of Moving Pictures and Audio Systems amendment 4: Transport of Multiview Video over ITU-T Rec H.222.0 ISO/IEC 13818-1" ITU-T REC. H.222.0(May 2006)FPDAM 4, vol. MPEG2009, No. 10572, May 11, 2009, pp. 1-20, XP002605067 p. 11, last two paragraphs sections 2.6.78 and 2.6.79 table T-1.

Shokrollahi, A.: "Raptor Codes," Internet Citation [Online] (Jan. 13, 2004) XP002367883, Retrieved from the Internet: URL: <http://www.cs.huji.ac.il/labs/danss/p2p/resources/raptor.pdf>.

Shokrollahi, Amin. "Raptor Codes," IEEE Transactions on Information Theory, Jun. 2006, vol. 52, No. 6, pp. 2551-2567, (search date: Feb. 1, 2010) URL: <<http://portal.acm.org/citation.cfm?id=1148681>> .

Shokrollahi et al., "Design of Efficient Erasure Codes with Differential Evolution", IEEE International Symposium on Information Theory, Jun. 25, 2000, pp. 5-5.

Sincoskie, W. D., "System Architecture for Large Scale Video on Demand Service," Computer Network and ISDN Systems, pp. 155-162, (1991).

Stockhammer, "WD 0.1 of 23001-6 Dynamic Adaptive Streaming over HTTP (DASH)", MPEG-4 Systems, International Organisation for Standardisation, ISO/IEC JTC1/SC29/WG11, Coding of Moving Pictures and Audio, MPEG 2010 Geneva/ml1398, Jan. 6, 2011, 16 pp.

Sullivan et al., Document: JVT-AA007, "Editors' Draft Revision to ITU-T Rec. H.264/ISO/IEC 14496-10 Advanced Video Coding—in Preparation for ITU-T SG 16 AAP Consent (in integrated form)," Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), 30th Meeting: Geneva, CH, Jan. 29-Feb. 3, 2009, pp. 1-683, [http://wftp3.itu.int/av-arch/jvt-site/2009\\_01\\_Geneva/JVT-AD007.zip](http://wftp3.itu.int/av-arch/jvt-site/2009_01_Geneva/JVT-AD007.zip).

Sun, et al., "Seamless Switching of Scalable Video Bitstreams for Efficient Streaming," IEEE Transactions on Multimedia, vol. 6, No. 2, Apr. 2004, pp. 291-303.

Telefon AB LM Ericsson, et al., "Media Presentation Description in httpstreaming", 3GPP Draft; S4-100080-MPD, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre; 650, Route Des Lucioles; F-06921 Sophia-Antipolis Cedex; France, vol. SA WG4, no. St Julians, Malta; 20100125, Jan. 20, 2010, XP050437773, [retrieved on Jan. 20, 2010].

Thomas Wiegand et al., "WD1: Working Draft 1 of High-Efficiency Video Coding", JCTVC-C403, Joint Collaborative Team on Video Coding (JCT-VC), of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 3rd Meeting: Guangzhou, CN, Oct. 7-15, 2010.

Todd, "Error Correction Coding: Mathematical Methods and Algorithms", Mathematical Methods and Algorithms, Jan. 1, 2005, pp. 451-534, Wiley, XP002618913.

Tsunoda T., et al., "Reliable Streaming Contents Delivery by Using Multiple Paths," Technical Report of the Institute of Electronics, Information and Communication Engineers, Japan, Mar. 2004, vol. 103, No. 692, pp. 187-190, NS2003-331, IN2003-286.

U.S. Appl. No. 12/840,146, by Ying Chen et al., filed Jul. 20, 2010.

U.S. Appl. No. 12/908,537, by Ying Chen et al., filed Oct. 20, 2010.

U.S. Appl. No. 12/908,593, by Ying Chen et al., filed Oct. 20, 2010.

U.S. Appl. No. 13/082,051, by Ying Chen et al., filed Apr. 7, 2011.

U.S. Appl. No. 13/205,559, by Ying Chen et al., filed Aug. 8, 2011.

U.S. Appl. No. 13/205,565, by Ying Chen et al., filed Aug. 8, 2011.

U.S. Appl. No. 13/205,574, by Ying Chen et al., filed Aug. 8, 2011.

Universal Mobile Telecommunications System (UMTS); LTE; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (3GPP TS 26.234 version 9.3.0 Release 9), Technical Specification, European Telecommunications Standards Institute (ETSI), 650, Route Des Lucioles; F-06921 Sophia-Antipolis; France, vol. 3GPP SA, No. V9.3.0, Jun. 1, 2010, XP014047290, paragraphs [5.5.4.2], [5.5.4.3], [5.5.4.4], [5.4.5], [5.5.4.6] paragraphs [10.2.3], [11.2.7], [12.2.3], [12.4.2], [12.6.2] paragraphs [12.6.3], [12.6.3.1], [12.6.4], [12.6.6].

Viswanathan, et al., "Metropolitan area video-on-demand services using pyramid broadcasting", Multimedia Systems, 4(4): 197-208 (1996).

Viswanathan, et al., "Pyramid Broadcasting for Video-on-Demand Service", Proceedings of the SPIE Multimedia Computing and Networking Conference, vol. 2417, pp. 66-77 (San Jose, CA, Feb. 1995).

Viswanathan, Subramaniam R., "Publishing in Wireless and Wireline Environments," Ph. D Thesis, Rutgers, The State University of New Jersey (Nov. 1994), 180 pages.

Watson, M., et al. "Asynchronous Layered Coding (ALC) Protocol Instantiation", IETF RFC 5775, pp. 1-23, (Apr. 2010).

(56)

## References Cited

## OTHER PUBLICATIONS

- Wiegand et al., "WD3: Working Draft 3 of High-Efficiency Video Coding," Document JCTVC-E603, 5th Meeting: Geneva, CH, Mar. 16-23, 2011, 193 pp.
- Wiegand T. et al., "WD2: Working Draft 2 of High-Efficiency Video Coding", 20110128, No. JCTVC-D503, Jan. 28, 2011, XP002679642, Retrieved from the Internet: URL: [http://wftp3.itu.int/av-arch/jctvc-site/2011\\_01\\_D\\_Daegu/](http://wftp3.itu.int/av-arch/jctvc-site/2011_01_D_Daegu/) [retrieved on Jul. 11, 2012].
- Wong, J.W., "Broadcast delivery", Proceedings of the IEEE, 76(12): 1566-1577, (1988).
- Yamanouchi N., et al., "Internet Multimedia Transmission with Packet by Using Forward Error Correction," Proceedings of DPS Workshop, the Information Processing Society of Japan, Dec. 6, 2000, vol. 2000, No. 15, pp. 145-150.
- Yamauchi, Nagamasa. "Application of Lost Packet Recovery by Front Error Correction to Internet Multimedia Transfer" Proceedings of Workshop for Multimedia Communication and Distributed Processing, Japan, Information Processing Society of Japan (IPS), Dec. 6, 2000, vol. 2000, No. 15, pp. 145-150.
- Yin et al., "Modified Belief-Propagation algorithm for Decoding of Irregular Low-Density Parity-Check Codes", Electronics Letters, IEE Stevenage, GB, vol. 38, No. 24, Nov. 21, 2002, pp. 1551-1553.
- Zorzi, et al.: "On the Statistics of Block Errors in Bursty Channels," IEEE Transactions on Communications, vol. 45, No. 6, Jun. 1997, pp. 660-667.
- 3GPP: "3rd Generation Partnership Project; Technical Specification Group Services and system Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (Release 6)", Sophia Antipolis, France, Jun. 1, 2005, XP002695256, Retrieved from the Internet: URL: [http://www.etsi.org/deliver/etsi\\_ts/126300\\_126399/126346/06.01.00\\_60/ts\\_126346v060100p.pdf](http://www.etsi.org/deliver/etsi_ts/126300_126399/126346/06.01.00_60/ts_126346v060100p.pdf).
- 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) (Release 10), 3GPP Standard; 3GPP TS 26.247, 3rd Generation Partnership Project (3GPP), Mobile Competence Centre ; 650, Route Des Lucioles ; F-06921 Sophia-Antipolis Cedex ; France, vol. SA WG4, No. V10.0.0, Jun. 17, 2011, pp. 1-94, XP050553206, [retrieved on Jun. 17, 2011].
- Anonymous: "Technologies under Consideration", 100. MPEG Meeting; Apr. 30, 2012-May 4, 2012; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11) No. N12682, Jun. 7, 2012, XP030019156.
- Anonymous: "Technologies under Consideration", 98. MPEG Meeting; Nov. 28, 2011-Dec. 2, 2011; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11) No. N12330, Dec. 3, 2011, XP030018825.
- Anonymous: "Text of ISO/IEC IS 23009-1 Media Presentation Description and Segment Formats", 98. MPEG Meeting; Nov. 28, 2011-Dec. 2, 2012; Geneva; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11), No. N12329, Jan. 6, 2012, XP030018824.
- Atis: "PTV Content on Demand Service", IIF-WT-063R44, Nov. 11, 2010, pp. 1-124, XP055045168, Retrieved from the Internet: URL: [ftp://vqeg.its.bldrdoc.gov/Documents/VQEG\\_Atlanta\\_Nov10/MeetingFiles/Liaison/IIF-WT-063R44\\_Content\\_on\\_Demand.pdf](ftp://vqeg.its.bldrdoc.gov/Documents/VQEG_Atlanta_Nov10/MeetingFiles/Liaison/IIF-WT-063R44_Content_on_Demand.pdf) [retrieved on Nov. 22, 2012].
- Bouazizi I., et al., "Proposals for ALC/FLUTE server file format (14496-12Amd.2)", 77. MPEG Meeting; Jul. 17, 2006-Jul. 21, 2006; Klagenfurt; (Motion Pictureexpert Group or ISO/IEC JTC1/SC29/WG11), No. M13675, Jul. 12, 2006, XP030042344, ISSN: 0000-0236.
- "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television; ETSI EN 300 744" ETSI Standards, LIS, Sophia Antipolis Cedex, France, V1.6.1, pp. 9, 10 Jan. 2009.
- Frojd P., et al., "Study on 14496-12:2005/PDAM2 ALU/ FLUTE Server File Format", 78. MPEG Meeting; Oct. 23, 2006-Oct. 27, 2006; Hangzhou: (Motion Pictureexpert Group or ISO/ IEC JTC1/SC29/WG11) No. M13855, Oct. 13, 2006, XP030042523, ISSN: 0000-0233.
- Gil A., et al., "Personalized Multimedia Touristic Services for Hybrid Broadcast/Broadband Mobile Receivers," IEEE Transactions on Consumer Electronics, 2010, vol. 56 (1), pp. 211-219.
- Hannuksela M.M., et al., "DASH: Indication of Subsegments Starting with SAP", 97. MPEG Meeting; Jul. 18, 2011-Jul. 22, 2011; Torino; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11) No. m21096, Jul. 21, 2011, XP030049659.
- Hannuksela M.M., et al., "ISO/BMFF: SAP definitions and 'sid' box", 97. MPEG Meeting; Jul. 18, 2011-Jul. 22, 2011; Torino; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11) No. m21435, Jul. 22, 2011, XP030049998.
- Kim J., et al., "Enhanced Adaptive Modulation and Coding Schemes Based on Multiple Channel Reportings for Wireless Multicast Systems", 62nd IEEE Vehicular Technology Conference, VTC-2005-FALL, Sep. 25-28, 2005, vol. 2, pp. 725-729, XP010878578, DOI: 10.1109/VETECE.2005.1558019, ISBN: 978-0-7803-9152-9.
- Li, M., et al., "Playout Buffer and Rate Optimization for Streaming over IEEE 802.11 Wireless Networks", Aug. 2009, Worcester Polytechnic Institute, USA.
- Luby et al., RaptorQ Forward Error Correction Scheme for Object Delivery draft-ietf-rmt-bb-fec-raptorq-00, Qualcomm, Inc. Jan. 28, 2010.
- Michael G et al., "Improved low-density parity-check codes using irregular graphs", Information Theory, IEEE Transactions on, Feb. 2001, vol. 47, No. 2, pp. 585-598.
- Moriyama, S., "5. Present Situation of Terrestrial Digital Broadcasting in Europe and USA", Journal of the Institute of Image Information and Television Engineers, Nov. 20, 1999, vol. 53, No. 11, pp. 1476-1478.
- Motorola et al: "An Analysis of DCD Channel Mapping to BCAS File Delivery Sessions; OMA-CD-DCD-2007-0112-INP\_DCD\_Channel\_Mapping\_to\_BCAST\_File\_Delivery", OMA-CD-DCD-2007-0112-INP\_DCD\_Channel\_Mapping\_to\_BCAST\_File\_Delivery, Open Mobile Alliance (OMA), 4330 La Jolla Village Dr., Suite 110 San Diego, CA 92122; USA Oct. 2, 2007, pp. 1-13, XP064036903.
- Ohashi A et al., "Low-Density Parity-Check (LDPC) Decoding of Quantized Data," Technical Report of the Institute of Electronics, Information and Communication Engineers, Aug. 23, 2002, vol. 102, No. 282, pp. 47-52, RCS2002-154.
- "RaptorQ Technical Overview", Qualcomm Incorporated, pp. 1-12 (Oct. 1, 2010).
- Roumy A., et al., "Unequal Erasure Protection and Object Bundle Protection with the Generalized Object Encoding Approach", Inria-00612583, Version 1, Jul. 29 2011, 25 pages.
- Schulzrinne, et al., "Real Time Streaming Protocol (RTSP)" Network Working Group, Request for Comments: 2326, Apr. 1998, pp. 1-92.
- Stockhammer T., et al., "DASH: Improvements on Representation Access Points and related flags", 97. MPEG Meeting; Jul. 18, 2011-Jul. 22, 2011; Torino; (Motion Picture Expert Group or ISO/IEC JTC1/SC29/WG11) No. m20339, Jul. 24, 2011, XP030048903.
- Wadayama T, "Introduction to Low Density Parity Check Codes and Sum-Product Algorithm," Technical Report of the Institute of Electronics, Information and Communication Engineers, Dec. 6, 2001, vol. 101, No. 498, pp. 39-46, MR2001-83.
- Yamazaki M., et al., "Multilevel Block Modulation Codes Construction of Generalized DFT," Technical Report of the Institute of Electronics, Information and Communication Engineers, Jan. 24, 1997, vol. 96, No. 494, pp. 19-24, IT96-50.
- Bross, et al., "High efficiency video coding (HEVC) text specification draft 6," JCTVC-H1003, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 8th Meeting: San Jose, CA, USA, Feb. 1-10, 2012, 259 pp.
- Makoto N., et al., "On Tuning of Blocking LU decomposition for VP2000 series" The 42th Information Processing Society of Japan Conference (1st term in 1991), Feb. 25, 1991, pp. 71-72, 4B-8B.
- Miller G., et al., "Bounds on the maximum likelihood decoding error probability of low density parity check codes", Information Theory, 2000. Proceedings. IEEE International Symposium on, 2000, p. 290.

(56)

**References Cited**

## OTHER PUBLICATIONS

Muramatsu J., et al., "Low density parity check matrices for coding of multiple access networks", Information Theory Workshop, 2003. Proceedings. 2003 IEEE, Apr. 4, 2003, pp. 304-307.

Samukawa, H. "Blocked Algorithm for LU Decomposition" Journal of the Information Processing Society of Japan, Mar. 15, 1993, vol. 34, No. 3, pp. 398-408.

3GPP TSG-SA4 #57 S4-100015, IMS based PSS and MBMS User Service extensions, Jan.19, 2010, URL: [http://www.3gpp.org/ftp/tsg\\_sa/ZWG4/Codec/TSGS4\\_57/docs/S4-100015.zip](http://www.3gpp.org/ftp/tsg_sa/ZWG4/Codec/TSGS4_57/docs/S4-100015.zip).

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (Release 9) 3GPP TS 26.234 V9.3.0, Jun. 23, 2010 P.85- 102, URL, [http://www.3gpp.org/ftp/TSG\\_SA/WG4\\_Codec/TSGS4\\_59/Docs/S4-100511.zip](http://www.3gpp.org/ftp/TSG_SA/WG4_Codec/TSGS4_59/Docs/S4-100511.zip), 26234-930.zip.

Lee, J.Y., "Description of Evaluation Experiments on ISO/IEC 23001-6, Dynamic Adaptive Streaming over HTTP", ISO/IEC JTC1/SC29/WG11MPEG2010/N11450, Jul. 31, 2010, 16 pp.

Luby M., "Simple Forward Error Correction (FEC) Schemes," draft-luby-rmt-bb-fec-supp-simple-00.txt, pp. 1-14, Jun. 2004.

Luby M., "LT Codes", Foundations of Computer Science, 2002, Proceedings, The 43rd Annual IEEE Symposium on, 2002.

Morioka S., "A Verification Methodology for Error Correction Circuits over Galois Fields", Tokyo Research Laboratory, IBM Japan Ltd, pp. 275-280, Apr. 22-23, 2002.

Qualcomm Incorporated: "Adaptive HTTP Streaming: Complete Proposal", 3GPP TSG-SA4 AHI Meeting S4-AHI170, Mar. 2, 2010, URL, [http://www.3gpp.org/FTP/tsg\\_sa/WG4\\_CODEC/Ad-hoc\\_MBS/Docs\\_AHI/S4-AHI170.zip](http://www.3gpp.org/FTP/tsg_sa/WG4_CODEC/Ad-hoc_MBS/Docs_AHI/S4-AHI170.zip), S4-AHI170\_CR\_AdaptiveHT-TPStreaming-Full.doc.

Qualcomm Incorporated: "Corrections to 3GPP Adaptive HTTP Streaming", 3GPP TSG-SA4 #59 Change Request 26.234 CR0172 S4-100403, Jun. 16, 2010, URL, [http://www.3gpp.org/FTP/tsg\\_sa/WG4\\_CODEC/TSGS4\\_59/Docs/54-100403.zip](http://www.3gpp.org/FTP/tsg_sa/WG4_CODEC/TSGS4_59/Docs/54-100403.zip), S4-100403\_CR\_26234-0172-AdaptiveHTTPStreaming-Rel-9.doc.

Gerard F., et al., "HTTP Streaming MPEG media—Response to CFP", 93. MPEG Meeting, Geneva Jul. 26, 2010 to Jul. 30, 2010.

Chikara S., et al., "Add-on Download Scheme for Multicast Content Distribution Using LT Codes", IEICE. B, Communications, Aug. 1, 2006, J89-B (8), pp. 1379-1389.

Hasan M.A., et al., "Architecture for a Low Complexity Rate-Adaptive Reed-Solomon Encoder", IEEE Transactions on Computers, IEEE Service Center, Los Alamitos, CA, US, vol. 44, No. 7, Jul. 1, 1995, pp. 938-942, XP000525729, ISSN: 0018-9340, DOI: 10.1109/12.392853.

Tetsuo M., et al., "Comparison of Loss Resilient Ability between Multi-Stage and Reed-Solomon Coding", Technical report of IEICE. CQ, Communication Quality, vol. 103 (178), Jul. 4, 2003, pp. 19-24. 1:811.

Watson M., et al., "Forward Error Correction (FEC) Framework draft-ietf-fecframe-framework-11," 2011, pp. 1-38, URL, <http://tools.ietf.org/pdf/draft-ietf-fecframe-framework-11.pdf>.

Watson M., et al., "Raptor FEC Schemes for FECFRAME draft-ietf-fecframe-raptor-04," 2010, pp. 1-21, URL, <http://tools.ietf.org/pdf/draft-ietf-fecframe-raptor-04.pdf>.

Qualcomm Incorporated: "RaptorQ Forward Error Correction Scheme for Object Delivery draft-ietf-rmt-bb-fec-raptor-04", Internet Engineering Task Force, IETF, pp. 1-68, Aug. 24, 2010.

Ramsey B, "HTTP Status: 206 Partial Content and Range Requests," May 5, 2008 obtained at <http://benramsey.com/blog/2008/05/206-partial-content-and-range-requests/>.

\* cited by examiner

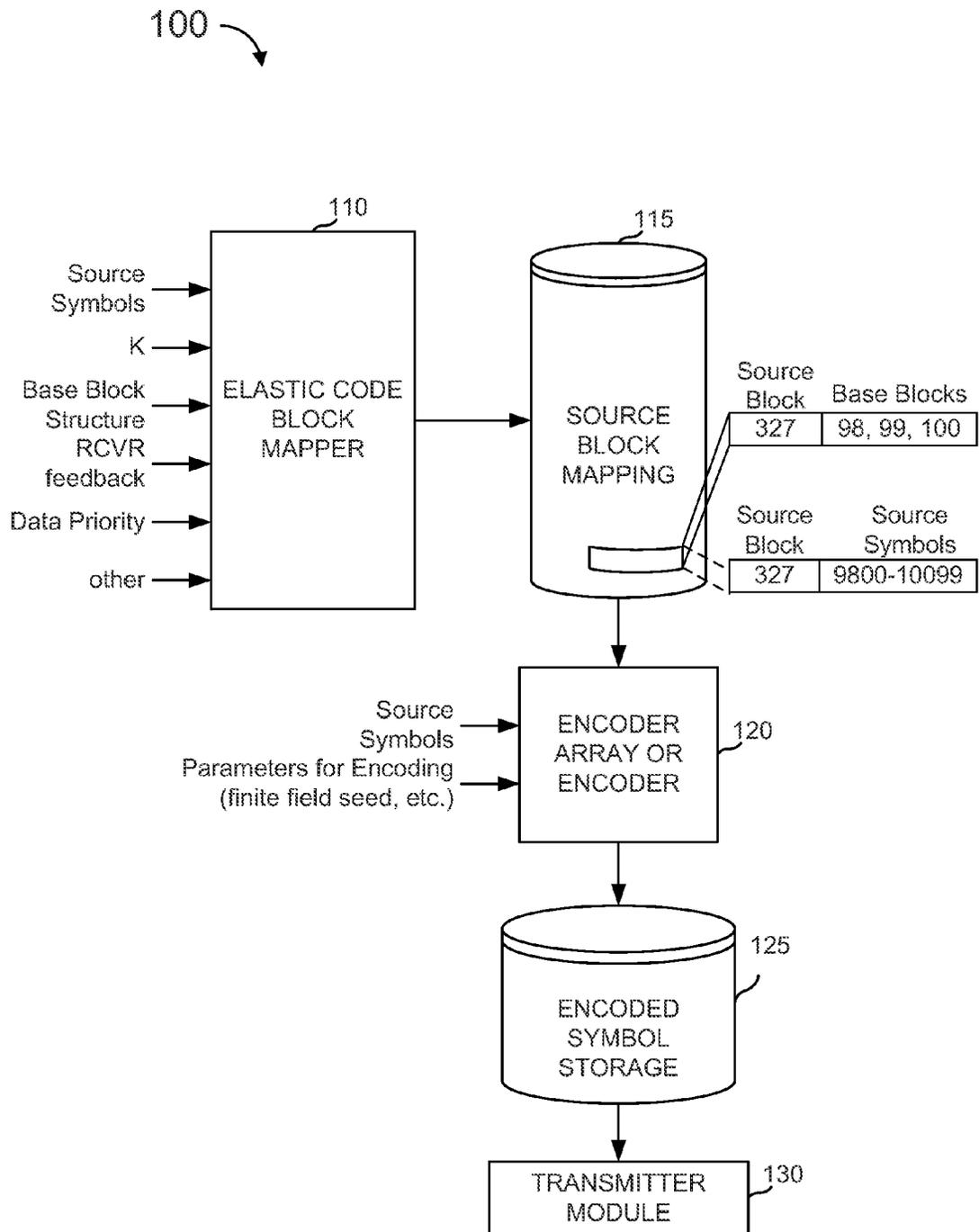


Figure 1

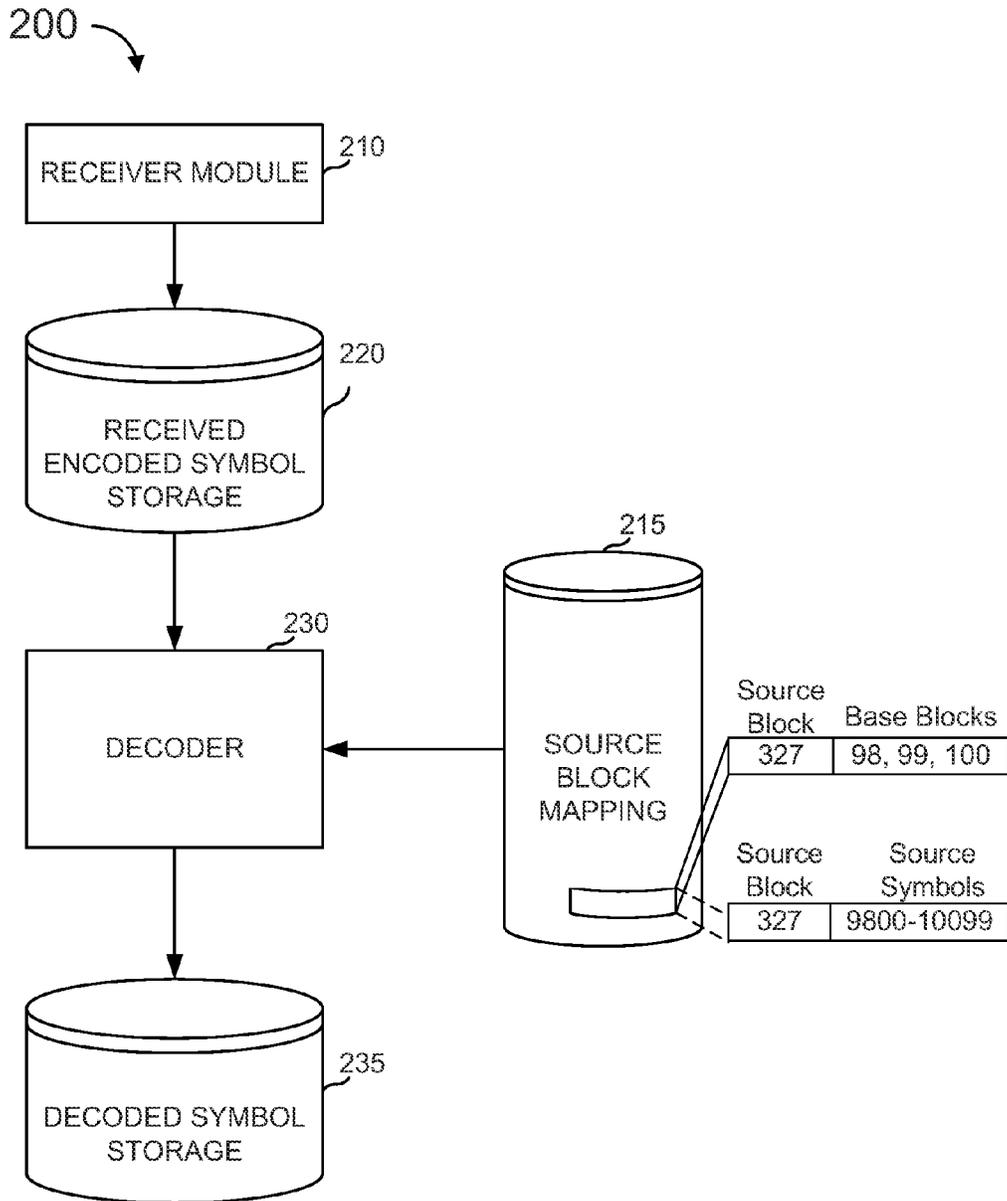


Figure 2

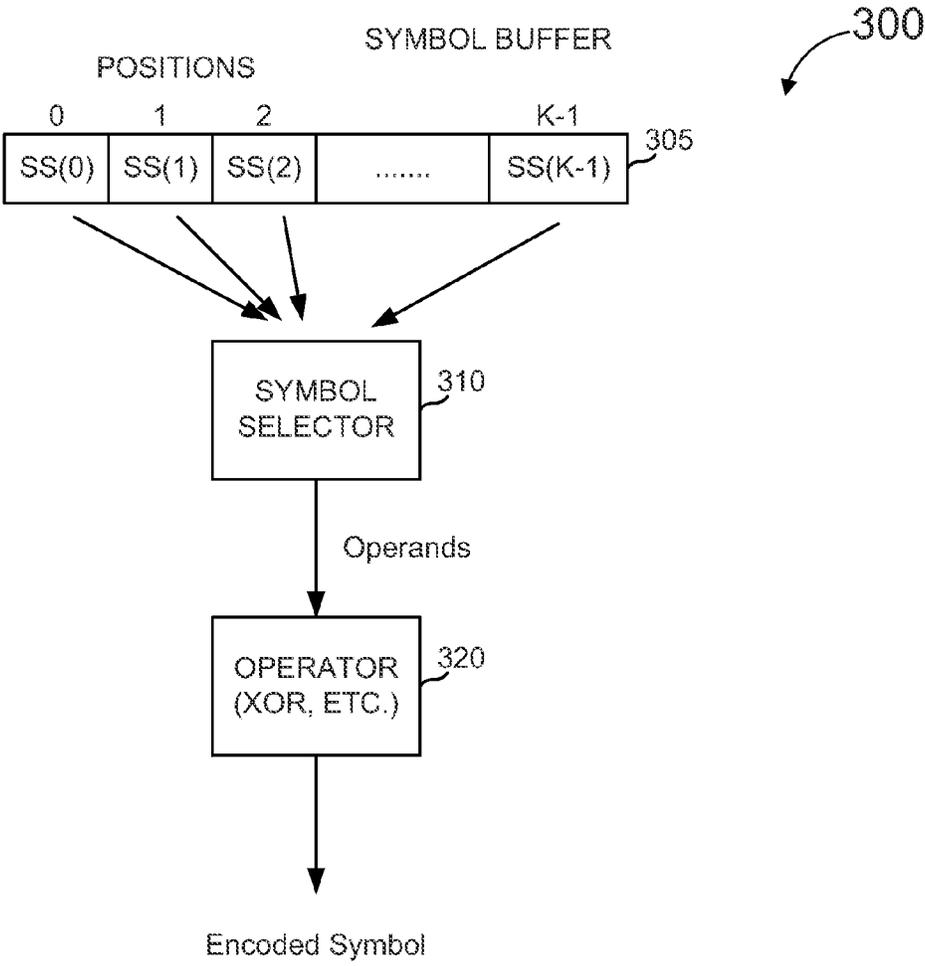


Figure 3

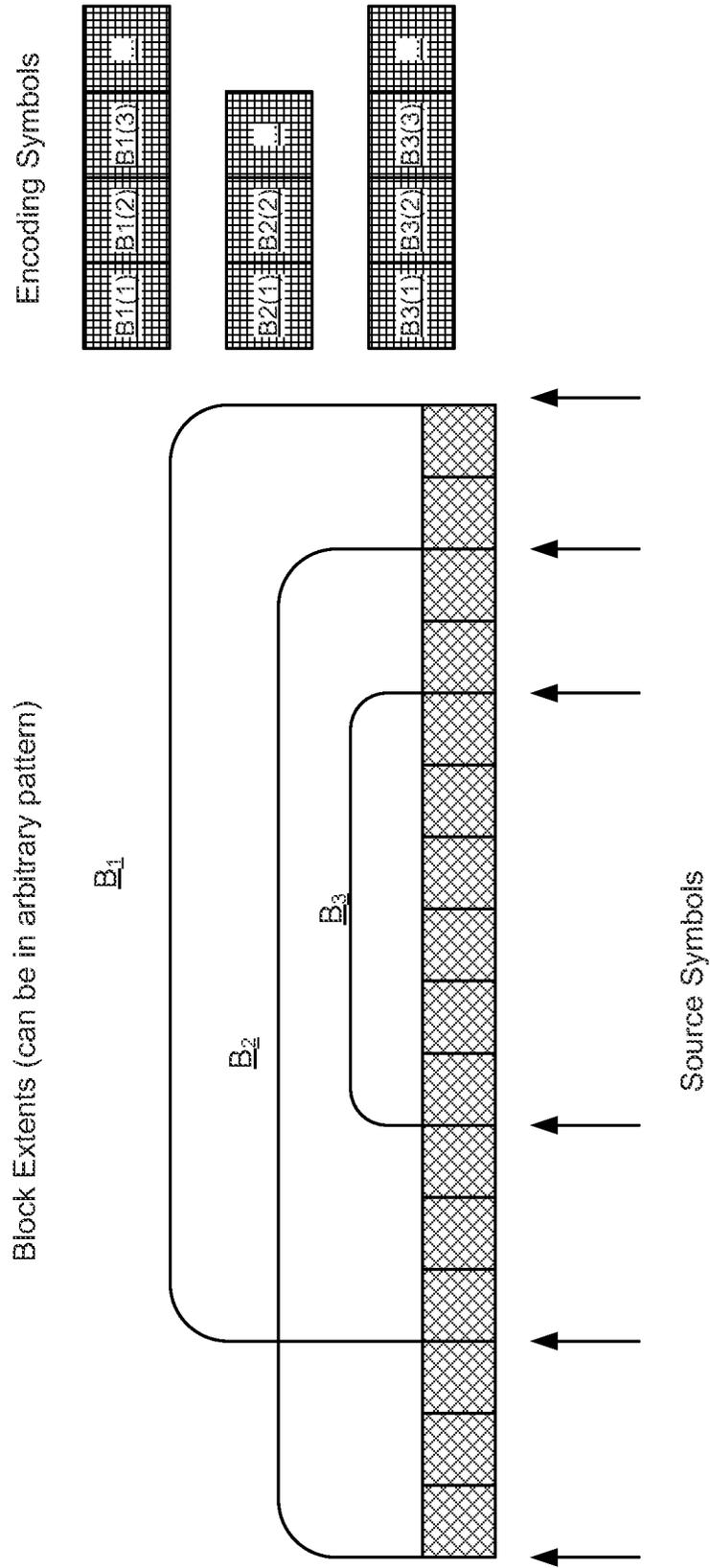


Figure 4

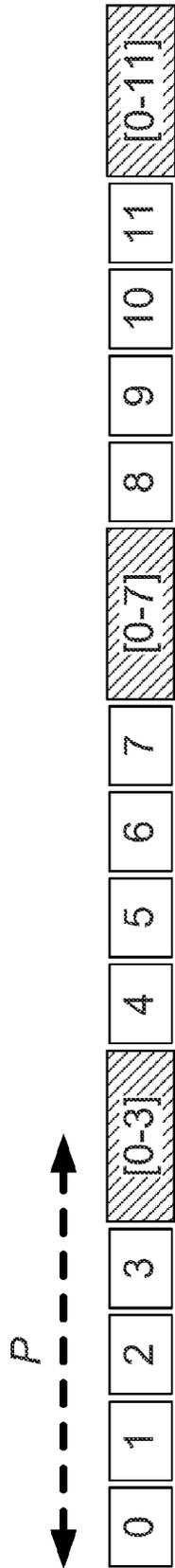


Figure 5

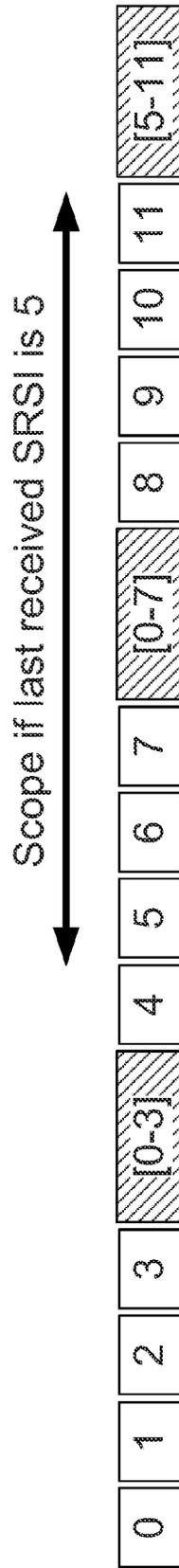


Figure 6

**ENCODING AND DECODING USING  
ELASTIC CODES WITH FLEXIBLE SOURCE  
BLOCK MAPPING**

CROSS REFERENCES

The Present Application for Patent is related to the following co-pending U.S. Patent Applications, each of which is filed concurrently herewith, assigned to the assignee hereof, and expressly incorporated by reference herein:

U.S. Patent Application entitled "Framing for an Improved Radio Link Protocol Including FEC" by Mark Watson, et al., having Ser. No. 13/025,925; and

U.S. Patent Application entitled "Forward Error Correction Scheduling for an Improved Radio Link Protocol" by Michael G. Luby, et al., having Ser. No. 13/025,934.

The following issued patents are expressly incorporated by reference herein for all purposes:

U.S. Pat. No. 6,909,383 entitled "Systematic Encoding and Decoding of Chain Reaction Codes" to Shokrollahi et al. issued Jun. 21, 2005 (hereinafter "Shokrollahi-Systematic"); and

U.S. Pat. No. 6,856,263 entitled "Systems and Processes for Decoding Chain Reaction Codes Through Inactivation" to Shokrollahi et al. issued Feb. 15, 2005 (hereinafter "Shokrollahi-Inactivation").

BACKGROUND

1. Field

The present disclosure relates in general to methods, circuits, apparatus and computer program code for encoding data for transmission over a channel in time and/or space and decoding that data, where erasures and/or errors are expected, and more particularly to methods, circuits, apparatus and computer program code for encoding data using source blocks that overlap and can be partially or wholly coextensive with other source blocks.

2. Background

Transmission of files between a sender and a recipient over a communications channel has been the subject of much literature. Preferably, a recipient desires to receive an exact copy of data transmitted over a channel by a sender with some level of certainty. Where the channel does not have perfect fidelity (which covers most all physically realizable systems), one concern is how to deal with data lost or garbled in transmission. Lost data (erasures) are often easier to deal with than corrupted data (errors) because the recipient cannot always tell when corrupted data is data received in error. Many error correcting codes have been developed to correct for erasures and/or for errors. Typically, the particular code used is chosen based on some information about the infidelities of the channel through which the data is being transmitted and the nature of the data being transmitted. For example, where the channel is known to have long periods of infidelity, a burst error code might be best suited for that application. Where only short, infrequent errors are expected a simple parity code might be best.

In particular applications, there is a need for handling more than one level of service. For example, a broadcaster might broadcast two levels of service, wherein a device capable of receiving only one level receives an acceptable set of data and a device capable of receiving the first level and the second level uses the second level to improve on the data of the first level. An example of this is FM radio, where some devices only received the monaural signal and others received that and the stereo signal. One characteristic of this scheme is that

the higher layers are not normally useful without the lower layers. For example, if a radio received the secondary, stereo signal, but not the base signal, it would not find that particularly useful, whereas if the opposite occurred, and the primary level was received but not the secondary level, at least some useful signal could be provided. For this reason, the primary level is often considered more worthy of protection relative to the secondary level. In the FM radio example, the primary signal is sent closer to baseband relative to the secondary signal to make it more robust.

Similar concepts exist in data transport and broadcast systems, where a first level of data transport is for a basic signal and a second level is for an enhanced layer. An example is H.264 Scalable Video Coding (SVC) wherein an H.264 base compliant stream is sent, along with enhancement layers. An example is a 1 megabit per second (mbps) base layer and a 1 mbps enhancement layer. In general, if a receiver is able to decode all of the base layer, the receiver can provide a useful output and if the receiver is able to decode all of the enhancement layer the receiver can provide an improved output, however if the receiver cannot decode all of the base layer, decoding the enhancement layer does not normally provide anything useful.

Forward error correction ("FEC") is often used to enhance the ability of a receiver to recover data that is transmitted. With FEC, a transmitter, or some operation, module or device operating for the transmitter, will encode the data to be transmitted such that the receiver is able to recover the original data from the transmitted encoded data even in the presence of erasures and or errors.

Because of the differential in the effects of loss of one layer versus another, different coding might be used for different layers. For example, the data for a base layer might be transmitted with additional data representing FEC coding of the data in the base layer, followed by the data of the enhanced layer with additional data representing FEC coding of the data in the base layer and the enhanced layer. With this approach, the latter FEC coding can provide additional assurances that the base layer can be successfully decoded at the receiver.

While such a layered approach might be useful in certain applications, it can be quite limiting in other applications. For example, the above approach can be impractical for efficiently decoding a union of two or more layers using some encoding symbols generated from one of the layers and other encoding symbols generated from the combination of the two or more layers.

SUMMARY

Data can be encoded by assigning source symbols to base blocks, assigning base blocks to source blocks and encoding each source block into encoding symbols, where at least one pair of source blocks is such they have at least one base block in common with both source blocks of the pair and at least one base block not in common with the other source block of the pair. The encoding of a source block can be independent of content of other source blocks. Decoding to recover all of a desired set of the original source symbols can be done from a set of encoding symbols from a plurality of source blocks wherein the amount of encoding symbols from the first source block is less than the amount of source data in the first source block and likewise for the second source block.

In specific embodiments, an encoder can encode source symbols into encoding symbols and a decoder can decode those source symbols from a suitable number of encoding symbols. The number of encoding symbols from each source

block can be less than the number of source symbols in that source block and still allow for complete decoding.

In a more specific embodiment where a first source block comprises a first base block and a second source block comprises the first base block and a second base block, a decoder can recover all of the first base block and second base block from a set of encoding symbols from the first source block and a set of encoding symbols from the second source block where the amount of encoding symbols from the first source block is less than the amount of source data in the first source block, and likewise for the second source block, wherein the number of symbol operations in the decoding process is substantially smaller than the square of the number of source symbols in the second source block.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a communications system that uses elastic codes according to aspects of the present invention.

FIG. 2 is a block diagram of an example of a decoder used as part of a receiver that uses elastic codes according to aspects of the present invention.

FIG. 3 illustrates, in more detail, an encoder, which might be the encoder shown in FIG. 1, or one encoder unit in an encoder array.

FIG. 4 illustrates an example of a source block mapping according to elastic codes.

FIG. 5 illustrates an elastic code that is a prefix code and  $G=4$ .

FIG. 6 illustrates an operation with a repair symbol's block.

Attached as Appendix A is a paper presenting Slepian-Wolf type problems on an erasure channel, with a specific embodiment of an encoder/decoder system, sometimes with details of the present invention used, which also includes several special cases and alternative solutions in some practical applications, e.g., streaming. It should be understood that the specific embodiments described in Appendix A are not limiting examples of the invention and that some aspects of the invention might use the teachings of Appendix A while others might not. It should also be understood that limiting statements in Appendix A may be limiting as to requirements of specific embodiments and such limiting statements might or might not pertain the claimed inventions and, therefore, the claim language need not be limited by such limiting statements.

To facilitate understanding, identical reference numerals have been used where possible to designate identical elements that are common to the figures, except that suffixes may be added, where appropriate, to differentiate such elements. The images in the drawings are simplified for illustrative purposes and are not necessarily depicted to scale.

The appended drawings illustrate exemplary configurations of the disclosure and, as such, should not be considered as limiting the scope of the disclosure that may admit to other equally effective configurations. Correspondingly, it has been contemplated that features of some configurations may be beneficially incorporated in other configurations without further recitation.

#### DETAILED DESCRIPTION

The present invention is not limited to specific types of data being transmitted. However in examples herein, it will be assumed that the data could be transmitted is represented by a sequence of one or more source symbols and that each

source symbol has a particular size, sometimes measured in bits. While it is not a requirement, in these examples, the source symbol size is also the size of encoding symbols. The "size" of a symbol can be measured in bits, whether or not the symbol is actually broken into a bit stream, where a symbol has a size of  $M$  bits when the symbol is selected from an alphabet of  $2^M$  symbols.

In the terminology used herein, the data to be conveyed is represented by a number of source symbols, where  $K$  is used to represent that number. In some cases,  $K$  is known in advance. For example, when the data to be conveyed is a file of unknown size and an integer multiple of the source symbol size,  $K$  would simply be the integer that is that multiple. However, it might also be the case that  $K$  is not known in advance of the transmission, or is not known until after the transmission has already started. For example, where the transmitter is transmitting a data stream as the transmitter receives the data and does not have an indication of when the data stream might end.

An encoder generates encoding symbols based on source symbols. Herein, the number of encoding symbols is often referred to as  $N$ . Where  $N$  is fixed given  $K$ , the encoding process has a code rate,  $r=K/N$ . Information theory holds that if all source symbol values are equally possible, perfect recovery of the  $K$  source symbols requires at least  $K$  encoding symbols to be received (assuming the same size for source symbols and encoding symbols) in order to fully recover the  $K$  source symbols. Thus, the code rate using FEC is usually less than one. In many instances, lower code rates allow for more redundancy and thus more reliability, but at a cost of lower bandwidth and possibly increased computing effort. Some codes require more computations per encoding symbol than others and for many applications, the computational cost of encoding and/or decoding will spell the difference between a useful implementation and an unwieldy implementation.

Each source symbol has a value and a position within the data to be transmitted and they can be stored in various places within a transmitter and/or receiver, computer-readable memory or other electronic storage, that contains a representation of the values of particular source symbols. Likewise, each encoding symbol has a value and an index, the latter being to distinguish one encoding symbol from another, and also can be represented in computer- or electronically-readable form. Thus, it should be understood that often a symbol and its physical representation can be used interchangeably in descriptions.

In a systematic encoder, the source symbols are part of the encoding symbols and the encoding symbols that are not source symbols are sometimes referred to as repair symbols, because they can be used at the decoder to "repair" damage due to losses or errors, i.e., they can help with recovery of lost source symbols. Depending on the codes used, the source symbols can be entirely recovered from the received encoding symbols which might be all repair symbols or some source symbols and some repair symbols. In a non-systematic encoder, the encoding symbols might include some of the source symbols, but it is possible that all of the encoding symbols are repair symbols. So as not to have to use separate terminology for systematic encoders and nonsystematic encoders, it should be understood that the term "source symbols" refers to symbols representing the data to be transmitted or provided to a destination, whereas the term "encoding symbols" refers to symbols generated by an encoder in order to improve the recoverability in the face of errors or losses, independent of whether those encoding symbols are source symbols or repair symbols. In some instances, the source symbols are preprocessed prior to presenting data to an

5

encoder, in which case the input to the encoder might be referred to as “input symbols” to distinguish from source symbols. When a decoder decodes input symbols, typically an additional step is needed to get to the source symbols, which is typically the ultimate goal of the decoder.

One efficient code is a simple parity check code, but the robustness is often not sufficient. Another code that might be used is a rateless code such as the chain reaction codes described in U.S. Pat. No. 6,307,487, to Luby, which is assigned to the assignee hereof, and expressly incorporated by reference herein (hereinafter “Luby I”) and the multi-stage chain reaction as described in U.S. Pat. No. 7,068,729, to Shokrollahi et al., which is assigned to the assignee hereof, and expressly incorporated by reference herein (hereinafter “Shokrollahi I”).

As used herein, the term “file” refers to any data that is stored at one or more sources and is to be delivered as a unit to one or more destinations. Thus, a document, an image, and a file from a file server or computer storage device, are all examples of “files” that can be delivered. Files can be of known size (such as a one megabyte image stored on a hard disk) or can be of unknown size (such as a file taken from the output of a streaming source). Either way, the file is a sequence of source symbols, where each source symbol has a position in the file and a value.

The term “file” might also, as used herein, refer to other data to be transmitted that is not be organized or sequenced into a linear set of positions, but may instead represent data may have orderings in multiple dimensions, e.g., planar map data, or data that is organized along a time axis and along other axes according to priorities, such as video streaming data that is layered and has multiple layers that depend upon one another for presentation.

Transmission is the process of transmitting data from one or more senders to one or more recipients through a channel in order to deliver a file. A sender is also sometimes referred to as the transmitter. If one sender is connected to any number of recipients by a perfect channel, the received data can be an exact copy of the input file, as all the data will be received correctly. Here, we assume that the channel is not perfect, which is the case for most real-world channels. Of the many channel imperfections, two imperfections of interest are data erasure and data incompleteness (which can be treated as a special case of data erasure). Data erasure occurs when the channel loses or drops data. Data incompleteness occurs when a recipient does not start receiving data until some of the data has already passed it by, the recipient stops receiving data before transmission ends, the recipient chooses to only receive a portion of the transmitted data, and/or the recipient intermittently stops and starts again receiving data.

If a packet network is used, one or more symbol, or perhaps portions of symbols, are included in a packet for transmission and each packet is assumed to have been correctly received or not at all. A transmission can be “reliable”, in that the recipient and the sender will correspond with each other in the face of failures until the recipient satisfied with the result, or unreliable, in that the recipient has to deal with what is offered by the sender and thus can sometimes fail. With FEC, the transmitter encodes data, by providing additional information, or the like, to make up for information that might be lost in transit and the FEC encoding is typically done in advance of exact knowledge of the errors, attempting to prevent errors in advance.

In general, a communication channel is that which connects the sender and the recipient for data transmission. The communication channel could be a real-time channel, where the channel moves data from the sender to the recipient as the

6

channel gets the data, or the communication channel might be a storage channel that stores some or all of the data in its transit from the sender to the recipient. An example of the latter is disk storage or other storage device. In that example, a program or device that generates data can be thought of as the sender, transmitting the data to a storage device. The recipient is the program or device that reads the data from the storage device. The mechanisms that the sender uses to get the data onto the storage device, the storage device itself and the mechanisms that the recipient uses to get the data from the storage device collectively form the channel. If there is a chance that those mechanisms or the storage device can lose data, then that would be treated as data erasure in the communication channel.

An “erasure code” is a code that maps a set of  $K$  source symbols to a larger ( $>K$ ) set of encoding symbols with the property that the original source symbols can be recovered from some proper subsets of the encoding symbols. An encoder will operate to generate encoding symbols from the source symbols it is provided and will do so according to the erasure code it is provided or programmed to implement. If the erasure code is useful, the original source symbols (or in some cases, less than complete recovery but enough to meet the needs of the particular application) are recoverable from a subset of the encoding symbols that happened to be received at a receiver/decoder, if the subset is of size greater than or equal to the size of the source symbols (an “ideal” code), or at least this should be true with reasonably high probability. In practice, a “symbol” is usually a collection of bytes, possibly several hundred bytes, and all symbols (source and encoding) are the same size.

A “block erasure code” is an erasure code that maps one of a set of specific disjoint subsets of the source symbols (“blocks”) to each encoding symbol. When a set of encoding symbols is generated from one block, those encoding symbols can be used in combination with one another to recover that one block.

The “scope” of an encoding symbol is the block it is generated from and the block that the encoding symbol is used to decode, with other encoding symbols used in combination.

The “neighborhood set” of a given encoding symbol is the set of source symbols within the symbol’s block that the encoding symbol directly depends on. The neighborhood set might be a very sparse subset of the scope of the encoding symbol. Many block erasure codes, including chain reaction codes (e.g., LT codes), LDPC codes, and multi-stage chain reaction codes (e.g., Raptor codes), use sparse techniques to generate encoding symbols for efficiency and other reasons. One example of a measurement of sparseness is the ratio of the number of symbols in the neighborhood set that an encoding symbol depends on to the number of symbols in the block. For example, where a block comprises 256 source symbols ( $k=256$ ) and each encoding symbol is an XOR of between two and five of those 256 source symbols, the ratio would be between  $2/256$  and  $5/256$ . Similarly, where  $K=1024$  and each encoding symbol is a function of exactly three source symbols (i.e., each encoding symbol’s neighborhood set has exactly three members), then the ratio is  $3/1024$ .

For some codes, such as Raptor codes, encoding symbols are not generated directly from source symbols of the block, but instead from other intermediate symbols that are themselves generated from source symbols of the block. In any case, for Raptor codes, the neighborhood set can be much smaller than the size of the scope (which is equal to the number of source symbols in the block) of these encoding symbols. In these cases where efficient encoding and decoding is a concern and the resulting code construction is sparse,

the neighborhood set of an encoding symbol can be much smaller than its scope, and different encoding symbols may have different neighborhood sets even when generated from the same scope.

Since the blocks of a block erasure code are disjoint, the encoding symbols generated from one block cannot be used to recover symbols from a different block because they contain no information about that other block. Typically, the design of codes, encoders and decoders for such disjoint block erasure codes behave a certain way due to the nature of the code. If the encoders/decoders were simply modified to allow for non-disjoint blocks, i.e., where the scope of a block might overlap another block's scope, encoding symbols generated from the overlapping blocks would not be usable to efficiently recover the source symbols from the unions of the blocks, i.e., the decoding process does not allow for efficient usage of the small neighborhood sets of the encoding symbols when used to decode overlapping blocks. As a consequence, the decoding efficiency of the block erasure codes when applied to decode overlapping blocks is much worse than the decoding efficiency of these codes when applied to what they were designed for, i.e., decoding disjoint blocks.

A "systematic code" is one in which the set of encoding symbols contains the source symbols themselves. In this context, a distinction might be made between source symbols and "repair symbols" where the latter refers to encoding symbols other than those that match the source symbols. Where a systematic code is used and all of the encoding symbols are received correctly, the extras (the repair symbols) are not needed at the receiver, but if some source symbols are lost or erased in transit, the repair symbols can be used to repair such a situation so that the decoder can recover the missing source symbols. A code is considered to be "nonsystematic" if the encoding symbols comprise the repair symbols and source symbols are not directly part of the encoding symbols.

With these definitions in mind, various embodiments will now be described.

#### Overview of Encoders/Decoders for Elastic Codes

In an encoder, encoding symbols are generated from source symbols, input parameters, encoding rules and possibly other considerations. In the examples of block-based encoding described herein, this set of source symbols from which an encoding symbol could depend is referred to as a "source block", or alternatively, referred to as the "scope" of the encoding symbol. Because the encoder is block-based, a given encoding symbol depends only on source symbols within one source block (and possibly other details), or alternatively, depends only on source symbols within its scope, and does not depend on source symbols outside of its source block or scope.

Block erasure codes are useful for allowing efficient encoding, and efficient decoding. For example, once a receiver successfully recovers all of the source symbols for a given source block, the receiver can halt processing of all other received encoding symbols that encode for source symbols within that source block and instead focus on encoding symbols for other source blocks.

In a simple block erasure encoder, the source data might be divided into fixed-size, contiguous and non-overlapping source blocks, i.e., each source block has the same number of source symbols, all of the source symbols in the range of the source block are adjacent in locations in the source data and each source symbol belongs to exactly one source block. However, for certain applications, such constraints may lower performance, reduce robustness, and/or add to computational effort of encoding and/or decoding.

Elastic erasure codes are different from block erasure codes in several ways. One is that elastic erasure code encoders and decoders operate more efficiently when faced with unions of overlapping blocks. For some of the elastic erasure code methods described herein, the generated encoding symbols are sparse, i.e., their neighborhood sets are much smaller than the size of their scope, and when encoding symbols generated from a combination of scopes (blocks) that overlap are used to decode the union of the scopes, the corresponding decoder process is both efficient (leverages the sparsity of the encoding symbols in the decoding process and the number of symbol operations for decoding is substantially smaller than the number of symbol operations needed to solve a dense system of equations) and has small reception overhead (the number of encoding symbols needed to recover the union of the scopes might be equal to, or not much larger than, the size of the union of the scopes). For example, the size of the neighborhood set of each encoding symbol might be the square root of  $K$  when it is generated from a block of  $K$  source symbols, i.e., when it has scope  $K$ . Then, the number of symbol operations needed to recover the union of two overlapping blocks from encoding symbols generated from those two blocks might be much smaller than the square of  $K'$ , where the union of the two blocks comprises  $K'$  source symbols.

With the elastic erasure coding described herein, source blocks need not be fixed in size, can possibly include non-adjacent locations, as well as allowing source blocks to overlap such that a given source symbol is "enveloped" by more than one source block.

In embodiments of an encoder described below, the data to be encoded is an ordered plurality of source symbols and the encoder determines, or obtains a determination of, demarcations of "base blocks" representing source symbols such that each source symbol is covered by one base block and a determination and demarcation of source blocks, wherein a source block envelops one or more base blocks (and the source symbols in those base blocks). Where each source block envelops exactly one base block, the result is akin to a conventional block encoder. However, there are several useful and unexpected benefits in coding when the source blocks are able to overlap each other such that some base block might be in more than one source block such that two source blocks have at least one base block in their intersection and the union of the two source blocks includes more source symbols than are in either one of the source blocks.

If the encoding is such that the portion of the source data that is represented by the union of the pair of source blocks is recoverable from a combination of a first set of encoding symbols generated from the first source block of the pair and a second set of encoding symbols generated from the second source block of the pair, it can be possible to decode using fewer received symbols that might have been required if the more simple encoding process is used. In this encoding process, the resulting encoding symbols can, in some cases, be used in combination for efficient recovery of source symbols of more than one source block.

An illustration of why this is so is provided below, but first, examples of implementations will be described. It should be understood that these implementations can be done in hardware, program code executed by a processor or computer, software running on a general purpose computer, or the like. Elastic Code Ideal Recovery Property

For block codes, ideal recovery is the ability to recover the  $K$  source symbols of a block from any received set of  $K$  encoding symbols generated from the block. It is well-known that there are block codes with this ideal recovery property.

For example, Reed-Solomon codes used as erasure codes exhibit this ideal recovery property.

A similar ideal recovery property might be defined for elastic codes. Suppose an elastic code communications system is designed such that a receiver receives some set of encoding symbols (where the channel may have caused the loss of some of the encoding symbols, so the exact set might not be specifiable at the encoder) and the receiver attempts to recover all of the original source symbols, wherein the encoding symbols are generated at the encoder from a set of overlapping scopes. The overlapping scopes are such that the received encoding symbols are generated from multiple source blocks of overlapping source symbols, wherein the scope of each received encoding symbol is one of the source blocks. In other words, encoding symbols are generated from a set of T blocks (scopes)  $b_1, b_2, \dots, b_T$ , wherein each encoding symbol is generated from exactly one of the T blocks (scopes).

In this context, the ideal recovery property of an elastic erasure code can be described as the ability to recover the set of T blocks from a subset, E, of received encoding symbols, for any S such that  $1 \leq S \leq T$ , for all subsets  $\{i_1, \dots, i_s\}$ , of  $\{1, \dots, T\}$ , if the following holds: For all s such that  $1 \leq s \leq S$ , for all subsets  $\{i'_1, \dots, i'_s\}$  of  $\{i_1, \dots, i_s\}$ , the number of symbols in E generated from any of  $b_{i'_1}, \dots, b_{i'_s}$  is at most the size of the union of  $b_{i'_1}, \dots, b_{i'_s}$ , and the number of symbols in E generated from any of  $b_{i_1}, \dots, b_{i_s}$  is equal to the size of the union of  $b_{i_1}, \dots, b_{i_s}$ . Note that E may be a subset of the received encoding symbols, i.e., some received encoding symbols might not be considered when evaluating this ideal recovery definition to see if a particular set of blocks (scopes) are recoverable.

Ideally, recovery of a set of blocks (scopes) should be computationally efficient, e.g., the number of symbol operations that the decoding process uses might be linearly proportional to the number of source symbols in the union of the recovered scopes, as opposed to quadratic, etc.

It should be noted that, while some of the descriptions herein might describe methods and processes for elastic erasure code encoding, processing, decoding, etc. that, in some cases, achieve the ideal recovery properties described above, in other cases, only a close approximation of the ideal recovery and efficiency properties of elastic codes are achieved, while still being considered to be within the definitions of elastic erasure code encoding, processing, decoding, etc.

#### System Overview

FIG. 1 is a block diagram of a communications system 100 that uses elastic codes.

In system 100, an elastic code block mapper (“mapper”) 110 generates mappings of base blocks to source blocks, and possibly the demarcations of base blocks as well. As shown in FIG. 1, communications system 100 includes mapper 110, storage 115 for source block mapping, an encoder array or encoder 120, storage 125 for encoding symbols, and transmitter module 130.

Mapper 110 determines, from various inputs and possibly a set of rules represented therein, which source blocks will correspond with which base blocks and stores the correspondences in storage 115. If this is a deterministic and repeatable process, the same process can run at a decoder to obtain this mapping, but if it is random or not entirely deterministic, information about how the mapping occurs can be sent to the destination to allow the decoder to determine the mapping.

As shown, a set of inputs (by no means required to be exhaustive) are used in this embodiment for controlling the operation of mapper 110. For example, in some embodiments, the mapping might depend on the values of the source

symbols themselves, the number of source symbols (K), a base block structure provided as an input rather than generated entirely internal to mapper 110, receiver feedback, a data priority signal, or other inputs.

As an example, mapper 110 might be programmed to create source blocks with envelopes that depend on a particular indication of the base block boundaries provided as an input to mapper 110.

The source block mapping might also depend on receiver feedback. This might be useful in the case where receiver feedback is readily available to a transmitter and the receiver indicates successful reception of data. Thus, the receiver might signal to the transmitter that the receiver has received and recovered all source symbols up to an i-th symbol and mapper 110 might respond by altering source block envelopes to exclude fully recovered base blocks that came before the i-th symbol, which could save computational effort and/or storage at the transmitter as well as the receiver.

The source block mapping can depend on a data priority input that signals to mapper 110 varying data priority values for different source blocks or base blocks. An example usage of this is in the case where a transmitter is transmitting data and receives a signal that the data being transmitted is a lower priority than other data, in which case the coding and robustness can be increased for the higher priority data at the expense of the lower priority data. This would be useful, in applications such as map displays, where an end-user might move a “focus of interest” point as a map is loading, or in video applications where an end-user fast forwards or reverses during the transmission of a video sequence.

In any case, encoder array 120 uses the source block mapping along with the source symbol values and other parameters for encoding to generate encoding symbols that are stored in storage 125 for eventual transmission by transmitter module 130. Of course it should be understood that system 100 could be implemented entirely in software that reads source symbol values and other inputs and generates stored encoding symbols. Because the source block mapping is made available to the encoder array and encoding symbols can be independent of source symbols not in the source block associated with that encoding symbol, encoder array 120 can comprise a plurality of independently operating encoders that each operate on a different source block. It should also be understood that in some applications each encoding symbol is sent immediately or almost immediately after it is generated, and thus there might not be a need for storage 125, or an encoding symbol might be stored within storage 125 before it is transmitted for only a short duration of time.

Referring now to FIG. 2, an example of a decoder used as part of a receiver at a destination is shown. As illustrated there, a receiver 200 includes a receiver module 210, storage 220 for received encoding symbols, a decoder 230, storage 235 for decoded source symbols, and a counterpart source block mapping storage 215. Not shown is any connection needed to receive information about how to create the source block mapping, if that is needed from the transmitter.

Receiver module 210 receives the signal from the transmitter, possibly including erasures, losses and/or missing data, derives the encoding symbols from the received signal and stores the encoding symbols and storage 220.

Decoder 230 can read the encoding symbols that are available, the source block mapping from storage 215 to determine which symbols can be decoded from the encoding symbols based on the mappings, the available encoding symbols and the previously decoded symbols in storage 235. The results of decoder 230 can be stored in storage 235.

It should be understood that storage **220** for received encoded symbols and storage **235** for decoded source symbols might be implemented by a common memory element, i.e., wherein decoder **230** saves the results of decoding in the same storage area as the received encoding symbols used to decode. It should also be understood from this disclosure that encoding symbols and decoded source symbols may be stored in volatile storage, such as random-access memory (RAM) or cache, especially in cases where there is a short delay between when encoding symbols first arrive and when the decoded data is to be used by other applications. In other applications, the symbols are stored in different types of memory.

FIG. 3 illustrates in more detail an encoder **300**, which might be the encoder shown in FIG. 1, or one encoder unit in an encoder array. In any case, as illustrated, encoder **300** has a symbol buffer **305** in which values of source symbols are stored. In the illustration, all  $K$  source symbols are storable at once, but it should be understood that the encoder can work equally as well with a symbol buffer that has less than all of the source symbols. For example, a given operation to generate an encoding symbol might be carried out with symbol buffer only containing one source block's worth of source symbols, or even less than an entire source block's worth of source symbols.

A symbol selector **310** selects from one to  $K$  of the source symbol positions in symbol buffer **305** and an operator **320** operates on the operands corresponding to the source symbols and thereby generates an encoding symbol. In a specific example, symbol selector **310** uses a sparse matrix to select symbols from the source block or scope of the encoding symbols being generated and operator **320** operates on the selected symbols by performing a bit-wise exclusive or (XOR) operation on the symbols to arrive at the encoding symbols. Other operations besides XOR are possible.

As used herein, the source symbols that are operands for a particular encoding symbol are referred to as that encoding symbol's "neighbors" and the set of all encoding symbols that depend on a given source symbol are referred to as that source symbol's neighborhood.

When the operation is an XOR, a source symbol that is a neighbor of an encoding symbol can be recovered from that encoding symbol if all the other neighbors source symbols of that encoding symbol are available, simply by XORing the encoding symbol and the other neighbors. This may make it possible to decode other source symbols. Other operations might have like functionality.

With the neighbor relationships known, a graph of source symbols and encoding symbols would exist to represent the encoding relationships.

#### Details of Elastic Codes

Elastic codes have many advantages over either block codes or convolutional codes or network codes, and easily allow for what is coded to change based on feedback received during encoding. Block codes are limited due to the requirement that they code over an entire block of data, even though it may be advantageous to code over different parts of the data as the encoding proceeds, based on known error-conditions of the channel and/or feedback, taking into consideration that in many applications it is useful to recover the data in prefix order before all of the data can be recovered due to timing constraints, e.g., when streaming data.

Convolutional codes provide some protection to a stream of data by adding repair symbols to the stream in a predetermined patterned way, e.g., adding repair symbols to the stream at a predetermined rate based on a predetermined pattern. Convolutional codes do not allow for arbitrary source

block structures, nor do they provide the flexibility to generate varying amounts of encoding symbols from different portions of the source data, and they are limited in many other ways as well, including recovery properties and the efficiency of encoding and decoding.

Network codes provide protection to data that is transmitted through a variety of intermediate receivers, and each such intermediate receiver then encodes and transmits additional encoding data based on what it received. Network codes do not provide the flexibility to determine source block structures, nor are there known efficient encoding and decoding procedures that are better than brute force, and network codes are limited in many other ways as well.

Elastic codes provide a suitable level of data protection while at the same time allowing for real-time streaming experience, i.e., introducing as little latency in the process as possible given the current error conditions due to the coding introduced to protect against error-conditions.

As explained, an elastic code is a code in which each encoding symbol may be dependent on an arbitrary subset of the source symbols. One type of the general elastic code is an elastic chord code in which the source symbols are arranged in a sequence and each encoding symbol is generated from a set of consecutive source symbols. Elastic chord codes are explained in more detail below.

Other embodiments of elastic codes are elastic codes that are also linear codes, i.e., in which each encoding symbol is a linear sum of the source symbols on which it depends and a GF( $q$ ) linear code is a linear code in which the coefficients of the source symbols in the construction of any encoding symbol are members of the finite field GF( $q$ ).

Encoders and decoders and communications systems that use the elastic codes as described herein provide a good balance of minimizing latency and bandwidth overhead.

#### Elastic Code Uses for Multi-Priority Coding

Elastic codes are also useful in communications systems that need to deliver objects that comprise multiple parts for those parts may have different priorities of delivery, where the priorities are determined either statically or dynamically.

An example of static priority would be data that is partitioned into different parts to be delivered in a priority that depends on the parts, wherein different parts may be logically related or dependent on one another, in either time or some other causality dimension. In this case, the protocol might have no feedback from receiver to sender, i.e., be open-loop.

An example of dynamic priority would be a protocol that is delivering two-dimensional map information to an end user dynamically in parts as the end user focus on different parts of the map changes dynamically and unpredictably. In this case, the priority of the different parts of the map to be delivered changes based on unknown a-priori priorities that are only known based on feedback during the course of the protocol, e.g., in reaction to changing network conditions, receiver input or interest, or other inputs. For example, an end user may change their interest in terms of which next portion of the map to view based on information in their current map view and their personal inclinations and/or objectives. The map data may be partitioned into quadrants, and within each quadrant to different levels of refinement, and thus there might be a base block for each level of each quadrant, and source blocks might comprise unions of one or more base blocks, e.g., some source blocks might comprise unions of the base blocks associated with different levels of refinement within one quadrant, whereas other source blocks might comprise unions of base blocks associated with adjacent quadrants of one refinement level. This is an example of a closed-loop protocol.

## Encoders Using Elastic Erasure Coding

Encoders described herein use a novel coding that allows encoding over arbitrary subsets of data. For example, one repair symbol can encode over one set of data symbols while a second repair symbol can encode over a second set of data symbols, in such a way that the two repair symbols can recover from the loss of two source symbols in the intersections of their scopes, and each repair symbol can recover from the loss of one data symbol from the data symbols that is in their scope but not in the scope of the other repair symbol. One advantage of elastic codes is that they can provide an elastic trade-off between recovery capabilities and end-to-end latency. Another advantage of such codes is that they can be used to protect data of different priorities in such a way that the protection provided solely for the highest priority data can be combined with the data provided for the entire data to recover the entire data, even in the case when the repair provided for the highest priority data is not alone sufficient for recovery of the highest priority data.

These codes are useful in complete protocol designs in cases where there is no feedback and in cases where there is feedback within the protocol. In the case where there is feedback in the protocol, the codes can be dynamically changed based on the feedback to provide the best combination of provided protection and added latency due to the coding.

Block codes can be considered a degenerate case of using elastic codes, by having single source scopes—each source symbol belongs in only one source block. With elastic codes, source scope determination can be completely flexible, source symbols can belong to multiple source scopes, source scopes can be determined on the fly, in other than a pre-defined regular pattern, determined by underlying structure of source data, determined by transport conditions or other factors.

FIG. 4 illustrates an example, wherein the lower row of boxes represents source symbols and the bracing above the symbols indicates the envelope of the source blocks. In this example, there are three source blocks and thus there would be three encoded blocks, one that encodes for each one of the source blocks. In this example, if source blocks are formed from base blocks, there could be five base blocks with the base blocks demarcations indicated with arrows.

In general, encoders and decoders that use elastic codes would operate where each of the source symbols is within one base block but can be in more than one source block, or source scope, with some of the source blocks being overlapping and at least in some cases not entirely subsets of other source blocks, i.e., there are at least two source blocks that have some source symbols in common but also each have some source symbols present in one of the source blocks but not in the other. The source block is the unit from which repair symbols are generated, i.e., the scope of the repair symbols, such that repair symbols for one source block can be independent of source symbols not in that source block, thereby allowing the decoding of source symbols of a source block using encoded, received, and/or repair symbols of that source block without requiring a decoder to have access to encoded, received, or repair symbols of another source block.

The pattern of scopes of source blocks can be arbitrary, and/or can depend on the needs or requests of a destination decoder. In some implementations, source scope can be determined on-the-fly, determined by underlying structure of source data, determined by transport conditions, and/or determined by other factors. The number of repair symbols that can be generated from a given source block can be the same for each source block, or can vary. The number of repair symbols

generated from a given source block may be fixed based on a code rate or may be independent of the source block, as in the case of chain reaction codes.

In the case of traditional block codes or chain reaction codes, repair symbols that are used by the decoder in combination with each other to recover source symbols are typically generated from a single source block, whereas with the elastic codes described herein, repair symbols can be generated from arbitrary parts of the source data, and from overlapping parts of the source data, and the mapping of source symbols to source blocks can be flexible.

## Selected Design Considerations

Efficient encoding and decoding is primary concern in the design of elastic codes. For example, ideal efficiency might be found in an elastic code that can decode using a number of symbol operations that is linear in the number of recovered source symbols, and thus any decoder that uses substantially fewer symbol operations for recovery than brute force methods is preferable, where typically a brute force method requires a number of symbol operations that is quadratic in the number of recovered source symbols.

Decoding with minimal reception overhead is also a goal, where “reception overhead” can be represented as the number of extra encoding symbols, beyond what is needed by a decoder, that are needed to achieve the previously described ideal recovery properties. Furthermore, guaranteed recovery, or high probability recovery, or very high likelihood recovery, or in general high reliability recovery, are preferable. In other words, in some applications, the goal need not be complete recovery.

Elastic codes are useful in a number of environments. For example with layered coding, a first set of repair symbols is provided to protect a block of higher priority data, while a second set of repair symbols protects the combination of the higher priority data block and a block of lower priority data, requiring fewer symbols at decoding and if the higher priority data block was encoded separately and the lower priority data block was encoded separately. Some known codes provide for layered coding, but often at the cost of failing to achieve efficient decoding of unions of overlapping source blocks and/or failing to achieve high reliability recovery.

The elastic window-based codes described below can achieve efficient and high reliability decoding of unions of overlapping source blocks at the same time and can also do so in the case of layered coding.

## Combination with Network Coding

In another environment, network coding is used, where an origin node sends encoding of source data to intermediate nodes that may experience different loss patterns and intermediate nodes send encoding data generated from the portion of the encoding data that is received to destination nodes. The destination nodes can then recover the original source data by decoding the received encoding data received from multiple intermediate nodes. Elastic codes can be used within a network coding protocol, wherein the resulting solution provides efficient and high reliability recovery of the original source data.

## Simple Construction of Elastic Chord Codes

For the purposes of explanation, assume an encoder generates a set of repair symbols as follows, which provides a simple construction of elastic chord codes. This simple construction can be extended to provide elastic codes that are not necessarily elastic chord codes, in which case the identification of a repair symbol and its neighborhood set or scope is an extension of the identification described here. Generate an  $m \times n$  matrix,  $A$ , with elements in  $GF(256)$ . Denote the element in the  $i$ -th row and  $j$ -th column by  $A_{ij}$  and the source

symbols by  $S_j$  for  $j=0, 1, 2, \dots$ . Then, for any tuple  $(e, l, i)$ , where  $e, l$  and  $i$  are integers,  $e \geq l > 0$  and  $0 \leq i < m$  and a repair symbol  $R_{e,l,i}$  has a value as set out in Equation 1.

$$R_{e,l,i} = \sum_{j=e-l+1}^{j=e} A_{ij} S_{j \bmod n} \quad (\text{Eqn. 1})$$

Note that for  $R_{e,l,i}$  to be well-defined, a notion of multiplication of a symbol by an element of GF(256) and a notion of summation of symbols should be specified. In examples, herein, elements of GF(256) are represented as octets and each symbol, which can be a sequence of octets, is thought of as a sequence of elements of GF(256). Multiplication of a symbol by a field element entails multiplication of each element of the symbol by the same field element. Summation of symbols is simply the symbol formed from the concatenation of the sums of the corresponding field elements in the symbols to be summed.

The set of source symbols that appear in Equation 1 for a given repair symbol is known as the “scope” of the repair symbol, whereas the set of repair symbols that have a given source symbol appear in Equation 1 for each of those repair symbols is referred to as the “neighborhood” of the given source symbol. Thus, in this construction, the neighborhood set of a repair symbol is the same as the scope of the repair symbol.

The encoding symbols of the code then comprise the source symbols plus repair symbols, as defined herein, i.e., the constructed code is systematic.

Consider two alternative constructions for the matrix A, corresponding to two different elastic codes. For a “Random Chord Code”, the elements of A are chosen pseudo-randomly from the nonzero elements of GF(256). It should be understood herein throughout, unless otherwise indicated, where something is described as being chosen randomly, it should be assumed that pseudo-random selection is included in that description and, more generally, that random operations can be performed pseudo-randomly. For a “Cauchy Chord Code”, the elements of A are defined as shown in Equation 2, where  $k=255-m$ , and  $g(x)$  is the finite field element whose octet representation is  $x$ .

$$A_{ij} = (g(j \bmod k) \oplus g(255-i))^{-1} \quad (\text{Eqn. 2})$$

Decoding Symbols from an Encoding Using a Simple Construction of Elastic Chord Codes

As well as encoding symbols themselves, the decoder has access to identifying information for each symbol, which can just be an index, i.e., for a source symbol,  $S_j$ , the identifying information is the index,  $j$ . For a repair symbol,  $R_{e,l,i}$ , the identifying information is the triple  $(e, l, i)$ . Of course, the decoder also has access to the matrix A.

For each received repair symbol, a decoder determines the identifying information and calculates a value for that repair symbol from Equation 1 using source symbol values if known and the zero symbol if the source symbol value is unknown. When the value so calculated is added to the received repair symbol, assuming the repair symbol was received correctly, the result is a sum over the remaining unknown source symbols in the scope or neighborhood of the repair symbol.

For simplicity, this description has a decoder programmed to attempt to recover all unknown source symbols that are in the scope of at least one received repair symbol. Upon reading this disclosure, it should be apparent how to modify the decoder to recover less than all, or all with a high probability but less than certainty, or a combination thereof.

In this example, let  $t$  be the number of unknown source symbols that are in the union of the scopes of received repair symbols and let  $j_0, j_1, \dots, j_{t-1}$  be the indices of these unknown source symbols. Let  $u$  be the number of received repair symbols and denote the received repair symbols (arbitrarily) as  $R_0, \dots, R_{u-1}$ .

Construct the  $u \times t$  matrix E with entries  $E_{pq}$ , where  $E_{pq}$  is the coefficient of source symbol  $S_{j_q}$  in Equation 1 for repair symbol  $R_p$ , or zero if  $S_{j_q}$  does not appear in the equation. Then, if  $S = (S_{j_0}, \dots, S_{j_{t-1}})^T$  is a vector of the missing source symbols and  $R = (R_0, \dots, R_{u-1})^T$  is a vector of the received repair symbols after applying step 1, the expression in Equation 3 will be satisfied.

$$R = E \cdot S \quad (\text{Eqn. 3})$$

If E does not have rank  $u$ , then there exists a row of E that can be removed without changing the rank of E. Remove this, decrement  $u$  by one and renumber the remaining repair symbols so that Equation 3 still holds. Repeat this step until E has rank  $u$ .

If  $u=t$ , then complete decoding is possible, E is square, of full rank and therefore invertible. Since E is invertible, S can be found from  $E^{-1}R$ , and decoding is complete. If  $u < t$ , then complete decoding is not possible without reception of additional source and/or repair symbols of this subset of the source symbols or having other information about the source symbols from some other avenue.

If  $u < t$ , then let  $E'$  be a  $u \times u$  sub-matrix of E of full rank. With a suitable column permutation, E can be written as  $(E' | U)$ , where U is a  $u \times (t-u)$  matrix. Multiplying both sides of Equation 3 by  $E'^{-1}$ , the expression in Equation 4 can be obtained, which provides a solution for the source symbols corresponding to rows of  $E^{-1}R$  where  $E^{-1}U$  is zero.

$$E'^{-1}R = (I | E'^{-1}U) \cdot S \quad (\text{Eqn. 4})$$

Equation 4 allows simpler recovery of the remaining source symbols if further repair and/or source symbols are received.

Recovery of other portions of the source symbols might be possible even when recovery of all unknown source symbols that are in the scope of at least one received repair symbol is not possible. For example, it may be the case that, although some unknown source symbols are in the scope of at least one received repair symbol, there are not enough repair symbols to recover the unknown source symbols, or that some of the equations between the repair symbols and unknown source symbols are linearly dependent. In these cases, it may be possible to at least recover a smaller subset of the source symbols, using only those repair symbols with scopes that are within the smaller subset of source symbols.

Stream Based Decoder Using Simple Construction of Elastic Chord Codes

In a “stream” mode of operation, the source symbols form a stream and repair symbols are generated over a suffix of the source symbols at the time the repair is generated. This stream based protocol uses the simple construction of the elastic chord codes described above.

At the decoder, source and repair symbols arrive one by one, possibly with some reordering and as soon as a source or repair symbol arrives, the decoder can identify whether any lost source symbol becomes decodable, then decode and deliver this source symbol to the decoder’s output.

To achieve this, the decoder maintains a matrix  $(I | E'^{-1}U)$  and updates this each time a new source or repair symbol is received according to the procedures below.

Let  $D$  denote the “decoding matrix”,  $(IE^{t-1} \cdot U)$ . Let  $D_{ij}$  denote the element at position  $(i,j)$ ,  $D_{*j}$  denote the  $j$ -th column of  $D$  and  $D_{i*}$  denote the  $i$ -th row of  $D$ .

In the procedures described below, the decoder performs various operations on the decoding matrix. The equivalent operations are performed on the repair symbols to effect decoding. These could be performed concurrently with the matrix operations, but in some implementations, these operations are delayed until actual source symbols are recovered in the RecoverSymbols procedure described below.

Upon receipt of a source symbol, if the source symbol is one of the missing source symbols,  $S_{j*}$ , then the decoder removes the corresponding column of  $D$ . If the removed column was one of the first  $u$  columns, then the decoder identifies the repair symbol associated with the row that has a nonzero element in the removed column. The decoder then repeats the procedure described below for receipt of this repair symbol. If the removed column was not one of the first  $u$  columns, then the decoder performs the RecoverSymbols procedure described below.

Upon receipt of a repair symbol, first the decoder adds a new column to  $D$  for each source symbol that is currently unknown, within the scope of the new repair symbol and not already associated with a column of  $D$ . Next, the decoder adds a new row,  $D_{u*}$ , to  $D$  for the received repair symbol, populating this row with the coefficients from Equation 1.

For  $i$  from 0 to  $u-1$  inclusive, the decoder replaces  $D_{iu*}$  with  $(D_{iu*} - M_{iu} \cdot D_{i*})$ . This step results in the first  $u$  elements of  $D_{iu*}$  being eliminated (i.e., reduced to zero). If  $D_{uu}$  is nonzero after this elimination step, then the decoder performs column exchanges (if necessary) so that  $D_{uu}$  is nonzero and replaces  $D_{iu*}$  with  $(D_{uu}^{-1} \cdot D_{iu*})$ .

For  $i$  from  $u-1$  to 0 inclusive, the decoder replaces  $D_{i*}$  with  $(D_{i*} - D_{iu} \cdot D_{u*})$ . This step results in the elements of column  $u$  being eliminated (i.e., reduced to zero) except for row  $u$ .

The matrix is now once again in the form  $(IE^{t-1} \cdot U)$  and the decoder can set  $u := u + 1$ .

To perform the RecoverSymbols procedure, the decoder considers each row of  $E^{t-1} \cdot U$  that is zero, or for all rows of  $D$  if  $E^{t-1} \cdot U$  is empty. The source symbol whose column is nonzero in that row of  $D$  can be recovered. Recovery is achieved by performing the stored sequence of operations upon the repair symbols. Specifically, whenever the decoder replaces row  $D_{i*}$  with  $(D_{i*} - \alpha \cdot D_{j*})$ , it also replaces the corresponding repair symbol  $R_i$  with  $(R_i - \alpha \cdot R_j)$  and whenever row  $D_{i*}$  is replaced with  $(\alpha \cdot D_{j*})$ , it replaces repair symbol  $R_i$  with  $\alpha \cdot R_j$ .

Note that the order in which the operations are performed is important and are the same as the order in which the matrix operations were performed.

Once the operations have been performed, then for each row of  $E^{t-1} \cdot U$  that is zero, the corresponding repair symbol now has a value equal to that of the source symbol whose column is nonzero in that row of  $D$  and the symbol has therefore been recovered. This row and column can then be removed from  $D$ .

In some implementations, symbol operations are only performed when it has been identified that at least one symbol can be recovered. Symbol operations are performed for all rows of  $D$  but might not result in recovery of all missing symbols. The decoder therefore tracks which repair symbols have been “processed” and which have not and takes care to keep the processed symbols up-to-date as further matrix operations are performed.

A property of elastic codes, in this “stream” mode, is that dependencies may stretch indefinitely into the past and so the decoding matrix  $D$  may grow arbitrarily large. Practically, the implementation should set a limit on the size of  $D$ . In practical

applications, there is often a “deadline” for the delivery of any given source symbol—i.e., a time after which the symbol is of no use to the protocol layer above or after which the layer above is told to proceed anyway without the lost symbol.

The maximum size of  $D$  may be set based on this constraint. However, it may be advantageous for the elastic code decoder to retain information that may be useful to recover a given source symbol even if that symbol will never be delivered to the application. This is because the alternative is to discard all repair symbols with a dependency on the source symbol in question and it may be the case that some of those repair symbols could be used to recover different source symbols whose deadline has not expired.

An alternative limit on the size of  $D$  is related to the total amount of information stored in the elastic code decoder. In some implementations, received source symbols are buffered in a circular buffer and symbols that have been delivered are retained, as these may be needed to interpret subsequently received repair symbols (e.g., calculating values in Equation 1 above). When a source symbol is finally discarded (due to the buffer being full) it is necessary to discard (or process) any (unprocessed) repair symbols whose scope includes that symbol. Given this fact, and a source buffer size, perhaps the matrix  $D$  should be sized to accommodate the largest number of repair symbols expected to be received whose scopes are all within the source buffer.

An alternative implementation would be to construct the matrix  $D$  only when there was a possibility of successful decoding according to the ideal recovery property described above.

#### Computational Complexity

The computational complexity of the code described above is dominated by the symbol operations.

Addition of symbols can be the bitwise exclusive OR of the symbols. This can be achieved efficiently on some processors by use of wide registers (e.g., the SSE registers on CPUs following an x86 architecture), which can perform an XOR operation over 64 or 128 bits of data at a time. However, multiplication of symbols by a finite field element often must be performed byte-by-byte, as processors generally do not provide native instructions for finite field operations and therefore lookup tables must be used, meaning that each byte multiplication requires several processor instructions, including access to memory other than the data being processed.

At the encoder, Equation 1 above is used to calculate each repair symbol. This involves  $l$  symbol multiplications and  $l-1$  symbol additions, where  $l$  is the number of source symbols in the scope of the repair symbol. If each source symbol is protected by exactly  $r$  repair symbols, then the total complexity is  $O(r \cdot k)$  symbol operations, where  $k$  is the number of source symbols. Alternatively, if each repair symbol has a scope or neighborhood set of  $l$  source symbols, then the computational complexity per generated repair symbol is  $O(l)$  symbol operations. As used herein, the expression  $O(\cdot)$  should be understood to be the conventional “on the order of” function.

At the decoder, there are two components to the complexity: the elimination of received source symbols and the recovery of lost source symbols. The first component is equivalent to the encoding operation, i.e.,  $O(r \cdot k)$  symbol operations. The second component corresponds to the symbol operations resulting from the inversion of the  $u \times u$  matrix  $E$ , where  $u$  is the number of lost source symbols, and thus has complexity  $O(u^2)$  symbol operations.

For low loss rates,  $u$  is small and therefore, if all repair symbols are used at the decoder, encoding and decoding complexity will be similar. However, since the major compo-

ment of the complexity scales with the number of repair symbols, if not all repair symbols are used, then complexity should decrease.

As noted above, in an implementation, processing of repair symbols is delayed until it is known that data can be recovered. This minimizes the symbol operations and so the computational requirements of the code. However, it results in bursts of decoding activity.

An alternative implementation can smooth out the computational load by performing the elimination operations for received source symbols (using Equation 1) as symbols arrive. This results in performing elimination operations for all the repair symbols, even if they are not all used, which results in higher (but more stable) computational complexity. For this to be possible, the decoder must have information in advance about which repair symbols will be generated, which may not be possible in all applications.

#### Decoding Probability

Ideally, every repair symbol is either clearly redundant because all the source symbols in its scope are already recovered or received before it is received, or is useful for recovering a lost source symbol. How frequently this is true depends on the construction of the code.

Deviation from this ideal might be detected in the decoder logic when a new received repair symbol results in a zero row being added to D after the elimination steps. Such a symbol carries no new information to the decoder and thus is discarded to avoid unnecessary processing.

In the case of the random GF(256) code implementation, this may be to be the case for roughly 1 repair symbol in 256, based on the fact that when a new random row is added to a  $uxu+1$  matrix over GF(256) of full rank, the probability that the resulting  $uxu$  matrix does not have full rank is  $1/256$ .

In the case of the Cauchy code implementation, when used as a block code and where the total number of source and repair symbols is less than 256, the failure probability is zero. Such a code is equivalent to a Reed-Solomon code.

#### Block Mode Results

In tests of elastic chord codes used as a block code (i.e., generating a number of repair symbols all with scope equal to the full set of  $k$  source symbols), for fixed block size ( $k=256$ ) and repair amount ( $r=8$ ), encode speed and decode speed are about the same for varying block sizes above about 200 bytes, but below that, speed drops. This is likely because below 200 byte symbols (or some other threshold depending on conditions), the overhead of the logic required to determine the symbol operations is substantial compared to the symbol operations themselves, but for larger symbol sizes the symbol operations themselves are dominant.

In other tests, encoding and decoding speed as a function of the repair overhead ( $r/k$ ) for fixed block and symbol size showed that that encoding and decoding complexity is proportional to the number of repair symbols (and so speed is proportional to  $1/r$ ).

#### Stream Mode Results

When the loss rate is much less than the overhead, the average latency is low but it increases quickly as the loss rate approaches the code overhead. This is what one would expect because when the loss rate is much less than the overhead, then most losses can be recovered using a single repair symbol. As the loss rate increases, we more often encounter cases where multiple losses occur within the scope of a single repair symbol and this requires more repair symbols to be used.

Another fine-tuning that might occur is to consider the effect of varying the span of the repair symbols (the span is how many source symbols are in the scope or neighborhood set of the repair symbol), which was 256 in the examples

above. Reducing the span, for a fixed overhead, reduces the number of repair symbols that protect each source symbol and so one would expect this to increase the residual error rate. However, reducing the span also reduces the computational complexity at both encoder and decoder.

Window-based Code that is a Fountain Block Code

In many encoders and decoders, the amount of computing power and time allotted to encoding and decoding is limited. For example, where the decoder is in a battery-powered handheld device, decoding should be efficient and not require excessive computing power. One measure of the computing power needed for encoding and decoding operations is the number of symbol operations (adding two symbols, multiplying, XORing, copying, etc.) that are needed to decode a particular set of symbols. A code should be designed with this in mind. While the exact number of operations might not be known in advance, since it might vary based on which encoding symbols are received and how many encoding symbols are received, it is often possible to determine an average case or a worst case and configure designs accordingly.

This section describes a new type of fountain block code, herein called a "window-based code," that is the basis of some of the elastic codes described further below that exhibit some aspects of efficient encoding and decoding. The window-based code as first described is a non-systematic code, but as described further below, there are methods for transforming this into a systematic code that will be apparent upon reading this disclosure. In this case, the scope of each encoding symbol is the entire block of  $K$  source symbols, but the neighborhood set of each encoding symbol is much sparser, consisting of  $B \ll K$  neighbors, and the neighborhood sets of different encoding symbols are typically quite different.

Consider a block of  $K$  source symbols. The encoder works as follows. First, the encoder pads (logically or actually) the block with  $B$  zero symbols on each side to form an extended block of  $K+2B$  symbols,  $X_0, X_1, \dots, X_{K+2B-1}$ , i.e., the first  $B$  symbols and the last  $B$  symbols are zero symbols, and the middle  $K$  symbols are the source symbols. To generate an encoding symbol, the encoder randomly selects a start position,  $t$ , between 1 and  $K+B-1$  and chooses values  $\alpha_0, \dots, \alpha_{B-1}$  randomly or pseudo-randomly from a suitable finite field (e.g., GF(2) or GF(256)). The encoding symbol value,  $ESV$ , is then calculated by the encoder using the formula of Equation 5, in which case the neighborhood set of the generated encoding symbol is selected among the symbols in positions  $t$  through  $t+B-1$  in the extended block.

$$ESV = \sum_{j=0}^{B-1} \alpha_j \cdot X_{t+j} \quad (\text{Eqn. 5})$$

The decoder, upon receiving at least  $K$  encoding symbols, uses a to-and-fro sweep across the positions of the source symbols in the extended block to decode. The first sweep is from the source symbol in the first position to the source symbol in the last position of the block, matching that source symbol,  $s$ , with an encoding symbol,  $e$ , that can recover it, and eliminating dependencies on  $s$  of encoding symbols that can be used to recover source symbols in later positions, and adjusting the contribution of  $s$  to  $e$  to be simply  $s$ . The second sweep is from the source symbol in the last position to the source symbol in the first position of the block, eliminating dependencies on that source symbol  $s$  of encoding symbols used to recover source symbols in earlier positions. After a successful to-and-fro sweep, the recovered value of each source symbol is the value of the encoding symbol to which it is matched.

For the first sweep process, the decoder obtains the set,  $E$ , of all received encoding symbols. For each source symbol,  $s$ , in position  $i=B, \dots, B+K-1$  within the extended block, the

decoder selects the encoding symbol  $e$  that has the earliest neighbor end position among all encoding symbols in  $E$  that have  $s$  in their neighbor set and then matches  $e$  to  $s$  and deletes  $e$  from  $E$ . This selection is amongst those encoding symbols  $e$  for which the contribution of  $s$  to  $e$  in the current set of linear equations is non-zero, i.e.,  $s$  contributes  $\beta \cdot s$  to  $e$ , where  $\beta \neq 0$ . If there is no encoding symbol  $e$  to which the contribution of  $s$  is non-zero, then decoding fails, as  $s$  cannot be decoded. Once source symbol  $s$  is matched with an encoding symbol  $e$ , encoding symbol  $e$  is removed from the set  $E$ , Gaussian elimination is used to eliminate the contribution of  $s$  to all encoding symbols in  $E$ , and the contribution of  $s$  to  $e$  is adjusted to be simply  $s$  by multiplying  $e$  by the inverse of the coefficient of the contribution of  $s$  to  $e$ .

The second sweep process of the decoder works as follows. For each source symbol,  $s$ , in source position  $i=K-1, \dots, 0$ , Gaussian elimination is used to eliminate the contribution of  $s$  to all encoding symbols in  $E$  matched to source symbols in positions previous to  $i$ .

The decoding succeeds in fully recovering all the source symbols if and only if the system of linear equations defined by the received encoding symbols is of rank  $K$ , i.e., if the received encoding symbols have rank  $K$ , then the above decoding process is guaranteed to recover the  $K$  source symbols of the block.

The number of symbol operations per generated encoding symbol is  $B$ .

The reach of an encoding symbol is defined to be the set of positions within the extended block between the first position that is a neighbor of the encoding symbol and the last position that is a neighbor of the encoding symbol. In the above construction, the size of the reach of each encoding symbols is  $B$ . The number of decoding symbol operations is bounded by the sum of sizes of the reaches of the encoding symbols used for decoding. This is because, by the way the matching process described above is designed, an encoding symbol reach is never extended during the decoding process and each decoding symbol operation decreases the sum of the sizes of the encoding symbol reaches by one. This implies that the number of symbol operations for decoding the  $K$  source symbols is  $O(K \cdot B)$ .

There is a trade-off between the computational complexity of the window-based code and its recovery properties. It can be shown by a simple analysis that if  $B=O(K^{1/2})$  and if the finite field size is chosen to be large enough, e.g.,  $O(K)$ , then all  $K$  source symbols of the block can be recovered with high probability from  $K$  received encoding symbols, and the failure probability decreases rapidly as a function of each additionally received encoding symbol. The recovery properties of the window-based code are similar to those of a random GF[2] code or random GF[256] code when GF[2] or GF[256] are used as the finite field, respectively, and  $B=O(K^{1/2})$ .

A similar analysis can be used to show that if  $B=O(\ln(K/\delta)/\epsilon)$  then all  $K$  source symbols of the block can be recovered with probability at least  $1-\delta$  after  $K \cdot (1+\epsilon)$  encoding symbols have been received.

There are many variations of the window-based codes described herein, as one skilled in the art will recognize. As one example, instead of creating an extended block of  $K+2B$  symbols, instead one can generate encoding symbols directly from the  $K$  source symbols, in which case  $t$  is chosen randomly between 0 and  $K-1$  for each encoding symbol, and then the encoding symbol value is computed as shown in Equation 6. One way to decode for this modified window-based block code is to use a decoding procedure similar to that described above, except at the beginning a consecutive set of  $B$  of the  $K$  source symbols are "inactivated", the decoding

proceeds as described previously assuming that these  $B$  inactivated source symbol values are known, a  $B \times B$  system of equations between encoding symbols and the  $B$  inactivated source symbols is formed and solved, and then based on this and the results of the to-and-fro sweep, the remaining  $K-B$  source symbols are solved. Details of how this can work are described in Shokrollahi-Inactivation.

$$ESV = \sum_{j=0}^{B-1} \alpha_j X_{(t+j) \bmod K} \quad (\text{Eqn. 6})$$

#### Systematic Window-Based Block Code

The window-based codes described above are non-systematic codes. Systematic window-based codes can be constructed from these non-systematic window-based codes, wherein the efficiency and recovery properties of the so-constructed systematic codes are very similar to those of the non-systematic code from which they are constructed.

In a typical implementation, the  $K$  source symbols are placed at the positions of the first  $K$  encoding symbols generated by the non-systematic code, decoded to obtain an extended block, and then repair symbols are generated for the systematic code from the decoded extended block. Details of how this can work are described in Shokrollahi-Systematic. A simple and preferred such systematic code construction for this window-based block code is described below.

For the non-systematic window-based code described above that is a fountain block code, a preferred way to generate the first  $K$  encoding symbols in order to construct a systematic code is the following. Instead of choosing the start position  $t$  between 1 and  $K+B-1$  for the first  $K$  encoding symbols, instead do the following. Let  $B'=B/2$  (assume without loss of generality that  $B$  is even). Choose  $t=B', B'+1, \dots, B'+K-1$  for the first  $K$  encoding symbols. For the generation of the first  $K$  encoding symbols, the generation is exactly as described above, with the possible exception, if it is not already the case, that the coefficient  $\alpha_{B'}$  is chosen to be a non-zero element of the finite field (making this coefficient non-zero ensures that the decoding process can recover the source symbol corresponding to this coefficient from this encoding symbol). By the way that these encoding symbols are constructed, it is always possible to recover the  $K$  source symbols of the block from these first  $K$  encoding symbols.

The systematic code encoding construction is the following. Place the values of the  $K$  source symbols at the positions of the first  $K$  encoding symbols generated according to the process described in the previous paragraph of the non-systematic window-based code, use the to-and-fro decoding process of the non-systematic window-based code to decode the  $K$  source symbols of the extended block, and then generate any additional repair symbols using the non-systematic window-based code applied to the extended block that contains the decoded source symbols that result from the to-and-fro decoding process.

The mapping of source symbols to encoding symbols should use a random permutation of  $K$  to ensure that losses of bursts of consecutive source symbols (and other patterns of loss) do not affect the recoverability of the extended block from any portion of encoding symbols, i.e., any pattern and mix of reception of source and repair symbols.

The systematic decoding process is the mirror image of the systematic encoding process. Received encoding symbols are used to recover the extended block using the to-and-fro decoding process of the non-systematic window-based code, and then the non-systematic window-based encoder is applied to the extended block to encode any missing source symbols, i.e., any of the first  $K$  encoding symbols that are missing.

One advantage of this approach to systematic encoding and decoding, wherein decoding occurs at the encoder and encoding occurs at the decoder, is that the systematic symbols and the repair symbols can be created using a process that is consistent across both. In fact, the portion of the encoder that generates the encoding symbols need not even be aware that K of the encoding symbols will happen to exactly match the original K source symbols.

Window-Based Code that is a Fountain Elastic Code

The window-based code fountain block code can be used as the basis for constructing a fountain elastic code that is both efficient and has good recovery properties. To simplify the description of the construction, we describe the construction when there are multiple base blocks  $X^1, \dots, X^L$  of equal size, i.e., each of the L basic blocks comprise K source symbols. Those skilled in the art will recognize that these constructions and methods can be extended to the case when the basic blocks are not all the same size.

As described previously, a source block may comprise the union of any non-empty subset of the L base blocks. For example, one source block may comprise the first base block and a second source block may comprise the first and second base blocks and a third source block may comprise the second and third base blocks. In some cases, some or all of the base blocks have different sizes and some or all of the source blocks have different sizes.

The encoder works as follows. First, for each base block  $X^i$ , the encoder pads (logically or actually) the block with B zero symbols on each side to form an extended block of  $K+2B$  symbols  $X_0^i, X_1^i, \dots, X_{K+2B-1}^i$ , the first B symbols and the last B symbols are zero symbols, and the middle K symbols are the source symbols of base block  $X^i$ .

The encoder generates an encoding symbol for source block S as follows, where S comprises  $L'$  base blocks, and without loss of generality assume that these are the base blocks  $X^1, \dots, X^{L'}$ . The encoder randomly selects a start position, t, between 1 and  $K+B-1$  and for all  $i=1, \dots, L'$ , chooses values  $\alpha_0^i, \dots, \alpha_{B-1}^i$  randomly from a suitable finite field (e.g., GF(2) or GF(256)). For each  $i=1, \dots, L'$ , the encoder generates an encoding symbol value based on the same starting position t, i.e., as shown in Equation 7.

$$ESV^t = \sum_{j=0}^{B-1} \alpha_j^i \cdot X_{t+j}^i \quad (\text{Eqn. 7})$$

Then, the generated encoding symbol value ESV for the source block is simply the symbol finite field sum over  $i=1, \dots, L'$  of  $ESV^i$ , i.e., as shown in Equation 8.

$$ESV = \sum_{i=1, \dots, L'} ESV^i \quad (\text{Eqn. 8})$$

Suppose the decoder is used to decode a subset of the base blocks, and without loss of generality assume that these are the base blocks  $X^1, \dots, X^{L'}$ . To recover the source symbols in these  $L'$  base blocks, the decoder can use any received encoding symbol generated from source blocks that are comprised of a union of a subset of  $X^1, \dots, X^{L'}$ . To facilitate efficient decoding, the decoder arranges a decoding matrix, wherein the rows of the matrix correspond to received encoding symbols that can be used for decoding, and wherein the columns of the matrix correspond to the extended blocks for base blocks  $X^1, \dots, X^{L'}$  arranged in the interleaved order:

$$\begin{matrix} X_0^1, X_0^2, \dots, X_0^{L'}, X_1^1, X_1^2, \dots, X_1^{L'}, \dots, \\ X_{K+2B-1}^1, X_{K+2B-1}^2, \dots, X_{K+2B-1}^{L'} \end{matrix}$$

Similar to the previously described to-and-fro decoder for a fountain block code, the decoder uses a to-and-fro sweep

across the column positions in the above described matrix to decode. The first sweep is from the smallest column position to the largest column position of the matrix, matching the source symbol s that corresponds to that column position with an encoding symbol e that can recover it, and eliminating dependencies on s of encoding symbols that can be used to recover source symbols that correspond to later column positions, and adjusting the contribution of s to e to be simply s. The second sweep is from the largest column position to the smallest column position of the matrix from the source symbol in the last position to the source symbol in the first position of the block, eliminating dependencies on the source symbol s that corresponds to that column position of encoding symbols used to recover source symbols in earlier positions. After a successful to-and-fro sweep, the recovered value of each source symbol is the value of the encoding symbol to which it is matched.

For the first sweep process, the decoder obtains the set, E, of all received encoding symbols that can be useful for decoding base blocks  $X^1, \dots, X^{L'}$ . For each position  $i=L' \cdot B, \dots, L' \cdot (B+K) - 1$  that corresponds to source symbol s of one of the  $L'$  basic blocks, the decoder selects the encoding symbol e that has the earliest neighbor end position among all encoding symbols in E that have s in their neighbor set and then matches e to s and deletes e from E. This selection is amongst those encoding symbols e for which the contribution of s to e in the current set of linear equations is non-zero, i.e., s contributes  $\beta \cdot s$  to e, where  $\beta \neq 0$ . If there is no encoding symbol e to which the contribution of s is non-zero then decoding fails, as s cannot be decoded. Once source symbol s is matched with an encoding symbol e, encoding symbol e is removed from the set E, Gaussian elimination is used to eliminate the contribution of s to all encoding symbols in E, and the contribution of s to e is adjusted to be simply s by multiplying e by the inverse of the coefficient of the contribution of s to e.

The second sweep process of the decoder works as follows. For each position  $i=L' \cdot (B+K) - 1, \dots, L' \cdot B$  that corresponds to source symbol s of one of the  $L'$  basic blocks, Gaussian elimination is used to eliminate the contribution of s to all encoding symbols in E matched to source symbols corresponding to positions previous to i.

The decoding succeeds in fully recovering all the source symbols if and only if the system of linear equations defined by the received encoding symbols is of rank  $L' \cdot K$ , i.e., if the received encoding symbols have rank  $L' \cdot K$ , then the above decoding process is guaranteed to recover the  $L' \cdot K$  source symbols of the  $L'$  basic blocks.

The number of symbol operations per generated encoding symbol is  $B \cdot V$ , where V is the number of basic blocks enveloped by the source block from which the encoding symbol is generated.

The reach of an encoding symbol is defined to be the set of column positions between the smallest column position that corresponds to a neighbor source symbol and the largest column position that corresponds to a neighbor source symbol in the decoding matrix. By the properties of the encoding process and the decoding matrix, the size of the reach of an encoding symbol is at most  $B \cdot L'$  in the decoding process described above. The number of decoding symbol operations is at most the sum of the sizes of the reaches of the encoding symbols, as by the properties of the matching process described above, the reach of encoding symbols are never extended beyond their original reach by decoding symbol operations and each decoding symbol operation decreases the sum of the sizes of the encoding symbol reaches by one. This

implies that, and that the number of symbol operations for decoding the  $N=K \cdot L'$  source symbols in  $L'$  basic blocks is  $O(N \cdot B \cdot L')$ .

There is a trade-off between the computational complexity of the window-based code and its recovery properties. It can be shown by a simple analysis that if  $B=O(\ln(L) \cdot K^{1/2})$  and if the finite field size is chosen to be large enough, e.g.,  $O(L \cdot K)$ , then all  $L' \cdot K$  source symbols of the  $L'$  basic blocks can be recovered with high probability if the recovery conditions of an ideal recovery elastic code described previously are satisfied by the received encoding symbols for the  $L'$  basic blocks, and the failure probability decreases rapidly as a function of each additionally received encoding symbol. The recovery properties of the window-based code are similar to those of a random GF[2] code or random GF[256] code when GF[2] or GF[256] are used as the finite field, respectively, and  $B=O(\ln(L) \cdot K^{1/2})$ .

A similar analysis can be used to show that if  $B=O(\ln(L \cdot K / \delta) / \epsilon)$  then all  $L' \cdot K$  source symbols of the  $L'$  basic blocks can be recovered with probability at least  $1-\delta$  under the following conditions. Let  $T$  be the number of source blocks from which the received encoding symbols that are useful for decoding the  $L'$  basic blocks are generated. Then, the number of received encoding symbols generated from the  $T$  source blocks should be at least  $L' \cdot K \cdot (1+\epsilon)$ , and for all  $S \leq T$ , the number of encoding symbols generated from any set of  $S$  source blocks should be at most the number of source symbols in the union of those  $S$  source blocks.

The window-based codes described above are non-systematic elastic codes. Systematic window-based fountain elastic codes can be constructed from these non-systematic window-based codes, wherein the efficiency and recovery properties of the so-constructed systematic codes are very similar to those of the non-systematic code from which they are constructed, similar to the systematic construction described above for the window-based codes that are fountain block codes. Details of how this might work are described in Shokrollahi-Systematic.

There are many variations of the window-based codes described herein, as one skilled in the art will recognize. As one example, instead of creating an extended block of  $K+2B$  symbols for each basic block, instead one can generate encoding symbols directly from the  $K$  source symbols of each basic block that is part of the source block from which the encoding symbols is generated, in which case  $t$  is chosen randomly between 0 and  $K-1$  for each encoding symbol, and then the encoding symbol value is computed similar to that shown in Equation 6 for each such basic block.

One way to decode for this modified window-based block code is to use a decoding procedure similar to that described above, except at the beginning a consecutive set of  $L' \cdot B$  of the  $L' \cdot K$  source symbols are "inactivated", the decoding proceeds as described previously assuming that these  $L' \cdot B$  inactivated source symbol values are known, a  $L' \cdot B \times L' \cdot B$  system of equations between encoding symbols and the  $L' \cdot B$  inactivated source symbols is formed and solved, and then based on this and the results of the to-and-fro sweep, the remaining  $L' \cdot (K-B)$  source symbols are solved. Details of how this can work are described in Shokrollahi-Inactivation.

There are many other variations of the window-based code above. For example, it is possible to relax the condition that each basic block comprises the same number of source symbols. For example, during the encoding process, the value of  $B$  used for encoding each basic block can be proportional to the number of source symbols in that basic block. For example, suppose a first basic block comprises  $K$  source symbols and a second basic block comprises  $K'$  source symbols, and let  $\mu=K/K'$  be the ratio of the sizes of the blocks.

Then, the value  $B$  used for the first basic block and the corresponding value  $B'$  used for the second basic block can satisfy:  $B/B'=\mu$ . In this variation, the start position within the two basic blocks for computing the contribution of the basic blocks to an encoding symbol generated from a source block that envelopes both basic blocks might differ, for example the encoding process can choose a value  $\phi$  uniformly between 0 and 1 and then use the start position  $t=\phi \cdot (K+B-1)$  for the first basic block and use the start position  $t'=\phi \cdot (K'+B'-1)$  for the second basic block (where these values are rounded up to the nearest integer position). In this variation, when forming the decoding matrix at the decoder comprising the interleaved symbols from each of the basic blocks being decoded, the interleaving can be done in such a way that the frequency of positions corresponding to the first basic block to the frequency of positions corresponding to the second basic block is in the ratio  $\mu$ , e.g., if the first basic block is twice the size of the second basic block then twice as many column positions correspond to the first basic block as correspond to the second basic block, and this condition is true (modulo rounding errors) for any consecutive set of column positions within the decoding matrix.

There are many other variations as well, as one skilled in the art will recognize. For example, a sparse matrix representation of the decoding matrix can be used at the decoder instead of having to store and process the full decoding matrix. This can substantially reduce the storage and time complexity of decoding.

Other variations are possible as well. For example, the encoding may comprise a mixture of two types of encoding symbols: a majority of a first type of encoding symbols generated as described above and a minority of a second type of encoding symbols generated sparsely at random. For example the fraction of the first type of encoding symbols could be  $1-K^{-1/3}$  and the reach of each first type encoding symbol could be  $B=O(K^{1/3})$ , and the fraction of the second type of encoding symbols could be  $K^{-1/3}$  and the number of neighbors of each second type encoding symbol could be  $K^{2/3}$ . One advantage of such a mixture of two types of encoding symbols is that the value of  $B$  used for the first type to ensure successful decoding can be substantially smaller, e.g.,  $B=O(K^{1/3})$  when two types are used as opposed to  $B=O(K^{1/2})$  when only one type is used.

The decoding process is modified so that in a first step the to-and-fro decoding process described above is applied to the first type of encoding symbols, using inactivation decoding to inactivate source symbols whenever decoding is stuck to allow decoding to continue. Then, in a second step the inactivated source symbol values are recovered using the second type of encoding symbols, and then in a third step these solved encoding symbol values together with the results of the first step of the to-and-fro decoding are used to solve for the remaining source symbol values. The advantage of this modification is that the encoding and decoding complexity is substantially improved without degrading the recovery properties. Further variations, using more than two types of encoding symbols, are also possible to further improve the encoding and decoding complexity without degrading the recovery properties.

#### Ideal Recovery Elastic Codes

This section describes elastic codes that achieve the ideal recovery elastic code properties described previously. This construction applies to the case when the source blocks satisfy the following conditions: the source symbols can be arranged into an order such that the source symbols in each source block are consecutive, and so that, for any first source

block and for any second source block, the source symbols that are in the first source block but not in the second source block are either all previous to the second source block or all subsequent to the second source block, i.e., there is no first and second source blocks with some symbols of the first source block preceding the second source block and some symbols of the first source block following the second source block. For brevity, herein such codes are referred to as a No-Subset Chord Elastic code, or "NSCE code." NSCE codes include prefix elastic codes.

It should be understood that the "construction" herein may involve mathematical concepts that can be considered in the abstract, but that such constructions are applied to a useful purpose and/or for transforming data, electrical signals or articles. For example, the construction might be performed by an encoder that seeks to encode symbols of data for transmission to a receiver/decoder that in turn will decode the encodings. Thus, inventions described herein, even where the description focuses on the mathematics, can be implemented in encoders, decoders, combinations of encoders and decoders, processes that encoder and/or decode, and can also be implemented by program code stored on computer-readable media, for use with hardware and/or software that would cause the program code to be executed and/or interpreted.

In an example construction of an NSCE code, a finite field with  $n^{c(n)}$  field elements is used, where  $c(n)=O(n^C)$ , where  $C$  is the number of source blocks. An outline of the construction follows, and implementation should be apparent to one of ordinary skill in the art upon reading this outline. This construction can be optimized to further reduce the size of the needed finite field, at least somewhat, in some cases.

In the outline,  $n$  is the number of source symbols to be encoded and decoded,  $C$  is the number of source blocks, also called chords, used in the encoding process,  $c(n)$  is some predetermined value that is on the order of  $n^C$ . Since a chord is a subset (proper or not) of the  $n$  source symbols that are used in generating repair symbols and a "block" is a set of symbols generated from within the same domain, there is a one-to-one correspondence between the chords used and the blocks used. The use of these elements will now be described with reference to an encoder or a decoder, but it should be understood that similar steps might be performed by both, even if not explicitly stated.

An encoder will manage a variable,  $j$ , that can range from 1 to  $C$  and indicates a current block/chord being processed. By some logic or calculation, the encoder determines, for each block  $j$ , the number of source symbols,  $k_j$ , and the number of encoding symbols,  $n_j$ , associated with block  $j$ . The encoder can then construct a  $k_j \times n_j$  Cauchy matrix,  $M_j$ , for block  $j$ . The size of the field needed for the base finite field to represent the Cauchy matrices is thus the maximum of  $k_j + n_j$  over all  $j$ . Let  $q$  be the number of elements in this base field.

The encoder works over a larger field,  $F$ , with  $q^D$  elements, where  $D$  is on the order of  $q^C$ . Let  $\omega$  be an element of  $F$  that is of degree  $D$ . The encoder uses (at least logically) powers of  $\omega$  to alter the matrices to be used to compute the encoding symbols. For block 1 of the  $C$  blocks, the matrix  $M_1$  is left unmodified. For block 2, the row of  $M_2$  that corresponds to  $i$ -th source symbol is multiplied by  $\omega^i$ . For block  $j$ , the row of  $M_j$  that corresponds to  $i$ -th source symbol is multiplied by  $\omega^{i \cdot q(j)}$ , where  $q(j)=q^{j-2}$ .

Let the modified matrices be  $M'_1, \dots, M'_C$ . These are the matrices used to generate the encoding symbols for the  $C$  blocks. A key property of these matrices flows from an observation explained below.

Suppose a receiver has received some mix of encoding symbols generated from the various blocks. That receiver

might want to determine whether the determinant of the matrix  $M$  corresponding to the source symbols and the received encoding symbols is nonzero.

Consider the bipartite graph between the received encoding symbols and the source symbols, with adjacencies defined naturally, i.e., there is an edge between an encoding symbol and a source symbol if the source symbol is part of the block from which the encoding symbol is generated. If there is a matching within this graph where all of the source symbols are matched, then the source symbols should be decodable from the received encoding symbols, i.e., the determinant of  $M$  should not be zero. Then, classify each matching by a "signature" of how the source symbols are matched to the blocks of encoding symbols, e.g., a signature of (1, 1, 3, 2, 3, 1, 2, 3) indicates that, in this matching, the first source symbol is matched to an encoding symbol in block 1, the second source symbol is matched to an encoding symbol in block 1, the third source symbol is matched to an encoding symbol in block 3, the fourth source symbol is matched to an encoding symbol in block 2, etc. Then, the matchings can be partitioned according to their signatures, and the determinant of  $M$  can be viewed as the sum of determinants of matrices defined by these signatures, where each such signature determinant corresponds to a Cauchy matrix and is thus not zero. However, the signature determinants could zero each other out.

By constructing the modified matrices  $M'_1, \dots, M'_C$ , a result is that there is a signature that uniquely has the largest power of  $\omega$  as a coefficient of the determinant corresponding to that signature, and this implies that the determinant of  $M$  is not zero since the determinant of this unique signature cannot be zeroed out by any other determinant. This is where the chord structure of the blocks is important.

Let the first block correspond to the chord that starts (and ends) first within the source symbols, and in general, let block  $j$  correspond to the chord that is the  $j$ -th chord to start (and finish) within the source blocks. Since there are no subset chords, if any one block starts before second one, it also has to end before the second one, otherwise the second one is a subset.

Then, the decoder handles a matching wherein all of the encoding symbols for the first block are matched to a prefix of the source symbols, wherein all of the encoding symbols for the second block are matched to a next prefix of the source symbols (excluding the source symbols matched to the first block), etc. In particular, this matching will have the signature of  $e_1$  1's followed by  $e_2$  2's, followed by  $e_3$  3's, etc., where  $e_i$  is the number of encoding symbols that are to be used to decode the source symbols that were generated from block  $i$ . This matching has a signature that uniquely has the largest power of  $\omega$  as a coefficient (similar to the argument used in the Theorem 1 for the two-chord case), i.e., any other signature that corresponds to a valid matching between the source and received encoding symbols will have a smaller power of  $\omega$  as a coefficient. Thus, the determinant has to be nonzero.

One disadvantage with chord elastic codes occurs where subsets exist, i.e., where there is one chord contained within another chord. In such cases, a decoder cannot be guaranteed to always find a matching where the encoding symbols for each block are used greedily, i.e., use all for block 1 on the first source symbols, followed by block 2, etc., at least according to the original ordering of the source symbols.

In some cases, the source symbols can be re-ordered to obtain the non-contained chord structure. For example, if the set of chords according to an original ordering of the source symbols were such that each subsequent chord contains all of the previous chords, then the source symbols can be re-ordered so that the structure is that of a prefix code, i.e., re-order

the source symbols from the inside to the out, so that the first source symbols are those inside all of the chords, followed by those source symbols inside all but the smallest chord, followed by those source symbols inside all but the smallest two chords, etc. With this re-ordering, the above constructions can be applied to obtain elastic codes with ideal recovery properties.

#### Examples of Usage of Elastic Codes

In one example, the encoder/decoder are designed to deal with expected conditions, such as a round-trip time (RTT) for packets of 400 ms, a delivery rate of 1 Mbps (bits/second), and a symbol size of 128 bytes. Thus, the sender sends approximately 1000 symbols per second ( $1000 \text{ symbols/sec} \times 128 \text{ bytes/symbol} \times 8 \text{ bits/byte} = 1.024 \text{ Mbps}$ ). Assume moderate loss conditions of some light loss (e.g., at most 5%) and sometimes heavier loss (e.g., up to 50%).

In one approach, a repair symbol is inserted after each  $G$  source symbols, and where the maximum latency can be as little as  $G$  symbols to recover from loss,  $X=1/G$  is the fraction of repair symbols that is allowed to be sent that may not recover any source symbols.  $G$  can change based on current loss conditions, RTT and/or bandwidth.

Consider the example in FIG. 5, where the elastic code is a prefix code and  $G=4$ . The source symbols are shown sequentially, and the repair symbols are shown with bracketed labels representing the source block that the repair symbol applies to.

If all losses are consecutive starting at the beginning, and one symbol is lost, then the introduced latency is at most  $G$ , whereas if two symbols are lost, then the introduced latency is at most  $2 \times G$ , and if  $i$  symbols are lost, the introduced latency is at most  $i \times G$ . Thus, the amount of loss affects introduced latency linearly.

Thus, if the allowable redundant overhead is limited to 5%, say, then  $G=20$ , i.e., one repair symbol is sent for each 20 source symbols. In the above example, one symbol is sent per 1 ms, so that would mean 20 ms between each repair symbol and the recovery time would be 40 ms for two lost symbols, 60 ms for three lost symbols, etc. Note that using just ARQ in these conditions, recovery time is at least 400 ms, the RTT.

In that example, a repair symbol's block is the set of all prior sent symbols. Where simple report back from the receiver are allowed, the blocks can be modified to exclude earlier source symbols that have been received or are no longer needed. An example is shown in FIG. 6, which is a variation of what is shown in FIG. 5.

In this example, assume that the encoder receives from the sender a SRSI indicator of the smallest Relevant Source Index. The SRSI can increase each time all prior source symbols are received or are no longer needed. Then, the encoder does not need to have any repair symbols depend on source symbols that have indices lower than the SRSI, which saves on computation. Typically, the SRSI is the index of the source symbol immediately following the largest prefix of already recovered source symbols. The sender then calculates scope of a repair symbol from the largest SRSI received from the receiver to the last sent index of a source symbol. This leads to exactly the same recovery properties as the no-feedback version, but lessens complexity/memory requirements at the sender and the receiver. In the example of FIG. 6,  $\text{SRSI}=5$ .

With the feedback, prefix elastic codes can be used more efficiently and feedback reduces complexity/memory requirements. When a sender gets feedback indicative of loss, it can adjust the scope of repair symbols accordingly. Thus, to combine forward error correction and reactive error correction, additional optimizations are possible. For example, the

forward error correction (FEC) can be tuned so that the allowable redundant overhead is high enough to proactively recover most losses, but not too high as to introduce too much overhead, while reactive correction is for the more rare losses. Since most losses are quickly recovered using FEC, most losses are recovered without an RTT latency penalty. While reactive correction has an RTT latency penalty, its use is rarer. Variations

Source block mapping indicates which blocks of source symbols are used for determining values for a set of encoding symbols (which can be encoding symbols in general or more specifically repair symbols). In particular, a source block mapping might be stored in memory and indicate the extents of a plurality of base blocks and indicate which of those base blocks are "within the scope" of which source blocks. In some cases, at least one base block is in more than one source block. In many implementations, the operation of an encoder or decoder can be independent of the source block mapping, thus allowing for arbitrary source block mapping. Thus, while predefined regular patterns could be used, that is not required and in fact, source block scopes might be determined from underlying structure of source data, by transport conditions or by other factors.

In some embodiments, an encoder and decoder can apply error-correcting elastic coding rather than just elastic erasure coding. In some embodiments, layered coding is used, wherein one set of repair symbols protects a block of higher priority data and a second set of repair symbols protects the combination of the block of higher priority data and a block of lower priority data.

In some communication systems, network coding is combined with elastic codes, wherein an origin node sends encoding of source data to intermediate nodes and intermediate nodes send encoding data generated from the portion of the encoding data that the intermediate node received—the intermediate node might not get all of the source data, either by design or due to channel errors. Destination nodes then recover the original source data by decoding the received encoding data from intermediate nodes, and then decodes this again to recover the source data.

In some communication systems that use elastic codes, various applications can be supported, such as progressive downloading for file delivery/streaming when prefix of a file/stream needs to be sent before it is all available, for example. Such systems might also be used for PLP replacement or for object transport.

Those of ordinary skill in the art would further appreciate, after reading this disclosure, that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the exemplary embodiments of the invention.

The various illustrative logical blocks, modules, and circuits described in connection with the embodiments disclosed herein may be implemented or performed with a general purpose processor, a Digital Signal Processor (DSP), an Application Specific Integrated Circuit (ASIC), a Field Pro-

grammable Gate Array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, or any combination thereof designed to perform the functions described herein. A general purpose processor may be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing devices, e.g., a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

The steps of a method or algorithm described in connection with the embodiments disclosed herein may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in Random Access Memory (RAM), flash memory, Read Only Memory (ROM), Electrically Programmable ROM (EPROM), Electrically Erasable Programmable ROM (EEPROM), registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. An exemplary storage medium is coupled to the processor such that the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor. The processor and the storage medium may reside in an ASIC. The ASIC may reside in a user terminal. In the alternative, the processor and the storage medium may reside as discrete components in a user terminal.

In one or more exemplary embodiments, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium. Computer-readable media includes both computer storage media and communication media including any medium that facilitates transfer of a computer program from one place to another. A storage media may be any available media that can be accessed by a computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if the software is transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-Ray™ disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

The previous description of the disclosed exemplary embodiments is provided to enable any person skilled in the art to make or use the present invention. Various modifications to these exemplary embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without departing from the spirit or scope of the invention. Thus, the present invention is not intended to be limited to the embodi-

ments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

The invention claimed is:

1. A method for encoding data to be transmitted from an electronic device or system to a receiver over a communications channel that could possibly introduce errors or erasures, wherein source data is represented by an ordered plurality of source symbols and the source data is recoverable from encoding symbols that are transmitted, the method comprising:

identifying a base block for each source symbol of the ordered plurality of source symbols, wherein the identified base block is one of a plurality of base blocks that, collectively, cover the source data to be encoded;

identifying, from a plurality of source blocks and for each base block, at least one source block that envelops that base block, wherein the plurality of source blocks includes at least one pair of source blocks that have a characteristic that there is at least one base block that is enveloped by both source blocks of the pair and at least one base block for each source block of the pair that is enveloped by that source block and not by the other source block of the pair;

encoding each of the plurality of source blocks according to an encoding process, resulting in encoding symbols, wherein the encoding process operates on one source block to generate encoding symbols, with the encoding symbols being independent of source symbol values of source symbols from base blocks not enveloped by the one source block, wherein the encoding is such that a portion of the source data that is represented by a union of the pair of source blocks is assured to be recoverable from a combination of a first set of encoding symbols generated from a first source block of the pair and a second set of encoding symbols generated from a second source block of the pair, wherein an amount of encoding symbols in the first set is less than an amount of source data in the first source block and an amount of encoding symbols in the second set is less than an amount of source data in the second source block; and  
outputting the encoding symbols for transmission to the receiver over the communications channel.

2. The method of claim 1, wherein the encoding process is such that, when the encoding symbols and the source symbols have a same size, when the first set of encoding symbols comprises  $M1$  encoding symbols, the first source block comprises  $N1$  source symbols, the second set of encoding symbols comprises  $M2$  encoding symbols, the second source block comprises  $N2$  source symbols, and when an intersection of the first and second source blocks comprises  $N3$  source symbols with  $N3$  greater than zero, then recoverability of the union of the pair of source blocks is assured beyond a predetermined threshold probability if  $M1+M2=N1+N2-N3$  for at least some combinations of values of  $M1 < N1$  and  $M2 < N2$ .

3. The method of claim 2, wherein the recoverability of the union of the pair of source blocks is assured beyond a predetermined threshold probability if  $M1+M2=N1+N2-N3$  for all combinations of values of  $M1$  and  $M2$  such that  $M1 \leq N1$  and  $M2 \leq N2$ .

4. The method of claim 2, wherein the recoverability of the union of the pair of source blocks is certain if  $M1+M2=N1+N2-N3$  for all combinations of values of  $M1$  and  $M2$  such that  $M1 \leq N1$  and  $M2 \leq N2$ .

5. The method of claim 2, wherein the recoverability of the union of the pair of source blocks is assured with a probability higher than a predetermined threshold probability if  $M1+M2$

is larger than  $N1+N2-N3$  by less than a predetermined percentage but smaller than  $N1+N2$  for at least some combinations of values of  $M1$  and  $M2$ .

6. The method of claim 1, wherein at least one encoding symbol generated from a source block is equal to a source symbol from a portion of the source data that is represented by that source block.

7. The method of claim 1, wherein the encoding process is such that a portion of the source data that is represented by the first source block of the pair is assured to be recoverable from a third set of encoding symbols generated from the first source block, wherein an amount of encoding symbols in the third set is no greater than the amount of source data in the first source block.

8. The method of claim 1, wherein the encoding process is such that a portion of the source data that is represented by the first source block of the pair is assured to be recoverable with a probability higher than a predetermined threshold probability from a third set of encoding symbols generated from the first source block, wherein an amount of encoding symbols in the third set is only slightly greater than the amount of source data in the first source block.

9. The method of claim 1, wherein a number of distinct encoding symbols that can be generated from each source block is independent of a size of the source block.

10. The method of claim 1, wherein a number of distinct encoding symbols that can be generated from each source block depends on a size of the source block.

11. The method of claim 1, wherein identifying base blocks for source symbols is performed prior to a start to encoding.

12. The method of claim 1, wherein identifying source blocks for base blocks is performed prior to a start to encoding.

13. The method of claim 1, wherein at least one encoding symbol is generated before a base block is identified for each source symbol or before the enveloped base blocks are determined for each of the source blocks or before all of the source data is generated or made available.

14. The method of claim 1, further comprising:  
receiving receiver feedback representing results at a decoder that is receiving or has received encoding symbols; and  
adjusting one or more of membership of source symbols in base blocks, membership of base blocks in enveloping source blocks, number of source symbols per base block, number of symbols in a source block, and/or number of encoding symbols generated from a source block, wherein the adjusting is done based on, at least in part, the receiver feedback.

15. The method of claim 14, wherein adjusting includes determining new base blocks or changing membership of source symbols in previously determined base blocks.

16. The method of claim 14, wherein adjusting includes determining new source blocks or changing envelopment of base blocks for previously determined source blocks.

17. The method of claim 1, further comprising:  
receiving data priority preference signals representing varying data priority preferences over the source data; and  
adjusting one or more of membership of source symbols in base blocks, membership of base blocks in enveloping source blocks, number of source symbols per base block, number of symbols in a source block, and/or number of encoding symbols generated from a source block, wherein the adjusting is done based on, at least in part, the data priority preference signals.

18. The method of claim 1, wherein a number of source symbols in the base blocks enveloped by each source block is independent, as between two or more of the source blocks.

19. The method of claim 1, wherein source symbols identified to a base block are not consecutive within the ordered plurality of source symbols.

20. The method of claim 1, wherein source symbols identified to a base block are consecutive within the ordered plurality of source symbols.

21. The method of claim 20, wherein source symbols identified to the base blocks enveloped by a source block are consecutive within the ordered plurality of source symbols.

22. The method of claim 1, wherein a number of encoding symbols that can be generated for a source block is independent of a number of encoding symbols that can be generated for other source blocks.

23. The method of claim 1, wherein a number of encoding symbols generated for a given source block is independent of a number of source symbols in base blocks enveloped by the given source block.

24. The method of claim 1, wherein encoding further comprises:

determining, for each encoding symbol, a set of coefficients selected from a finite field; and  
generating the encoding symbol as a combination of source symbols of one or more base blocks enveloped by a single source block, wherein the combination is defined, in part, by the set of coefficients.

25. The method of claim 1, wherein a number of symbol operations to generate an encoding symbol from a source block is linearly proportional to a number of source symbols in a portion of the source data that is represented by the source block.

26. A method for decoding data received at an electronic device or system and delivered from a transmitter over a communications channel that could possibly include errors or erasures, to recover source data that was represented by a set of source symbols, the method comprising:

identifying a base block for each source symbol, wherein the identified base block is one of a plurality of base blocks that, collectively, cover the source data;  
identifying, from a plurality of source blocks and for each base block, at least one source block that envelops that base block, wherein the plurality of source blocks includes at least one pair of source blocks that have a characteristic that there is at least one base block that is enveloped by both source blocks of the pair and at least one base block for each source block of the pair that is enveloped by that source block and not by the other source block of the pair; and  
receiving a plurality of received symbols;

for each received symbol, identifying a source block for which that received symbol is an encoding symbol for;  
decoding a set of source symbols from the plurality of received symbols, wherein a portion of the source data that is represented by a union of the pair of source blocks is assured to be recoverable from a combination of a first set of received symbols corresponding to encoding symbols that were generated from a first source block of the pair and a second set of received symbols corresponding to encoding symbols that were generated from a second source block of the pair, wherein an amount of received symbols in the first set is less than an amount of source data in the first source block and an amount of received symbols in the second set is less than an amount of source data in the second source block; and

35

outputting the decoded source symbols in a computer-readable form.

27. The method of claim 26, wherein if  $N1$  is a number of source symbols in source data of the first source block, if  $N2$  is a number of source symbols in source data of the second source block, if  $N3$  is a number of source symbols in an intersection of the first and second source blocks with  $N3$  greater than zero, if the encoding symbols and the source symbols have a same size, if  $R1$  is a number of received symbols in the first set of received symbols, if  $R2$  is a number of received symbols in the second set of received symbols, then decoding the union of the pair of source blocks from the first set of  $R1$  received symbols and from the second set of  $R2$  received symbols is assured beyond a predetermined threshold probability if  $R1+R2 \geq N1+N2-N3$ , for at least one value of  $R1$  and  $R2$  such that  $R1 < N1$  and  $R2 < N2$ .

28. The method of claim 27, wherein decoding the union of the pair of source blocks is assured beyond a predetermined threshold probability if  $R1+R2 = N1+N2-N3$  for all values of  $R1 \leq N1$  and  $R2 \leq N2$ .

29. The method of claim 27, wherein decoding the union of the pair of source blocks is certain if  $R1+R2 = N1+N2-N3$  for all values of  $R1 \leq N1$  and  $R2 \leq N2$ .

30. The method of claim 26, wherein a portion of the source data that is represented by the first source block of the pair is recoverable from a third set of encoding symbols generated from the first source block, wherein an amount of encoding symbols in the third set is no greater than the amount of source data in the first source block.

31. The method of claim 26, wherein a number of distinct encoding symbols that can be generated from each source block is independent of a size of the source block.

32. The method of claim 26, wherein at least one of identifying base blocks for source symbols and identifying source blocks for base blocks is performed prior to a start to decoding.

33. The method of claim 26, wherein at least some source symbols are decoded before a base block is identified for each source symbol and/or before the enveloped base blocks are determined for each of the source blocks.

34. The method of claim 26, further comprising:  
determining receiver feedback representing results at a decoder based on what received symbols have been received and/or what portion of the source data is desired at a receiver and/or data priority preference; and  
outputting the receiver feedback such that it is usable for altering an encoding process.

35. The method of claim 26, wherein a number of source symbols in the base blocks enveloped by each source block is independent, as between two or more of the source blocks.

36. The method of claim 26, wherein source symbols identified to a base block are consecutive within an ordered plurality of source symbols that is the set of source symbols.

37. The method of claim 26, wherein source symbols identified to the base blocks enveloped by a source block are consecutive within an ordered plurality of source symbols that is the set of source symbols.

38. The method of claim 26, wherein decoding further comprises:

determining, for each received symbol, a set of coefficients selected from a finite field; and  
decoding at least one source symbol from more than one received symbol or previously decoded source symbols using the set of coefficients for the more than one received symbol.

39. The method of claim 26, wherein a number of symbol operations to recover a union of one or more source blocks is

36

linearly proportional to a number of source symbols in a portion of the source data that is represented by the union of one or more source blocks.

40. An encoder of an electronic device or system that encodes data for transmission to a receiver over a communications channel that could possibly introduce errors or erasures, comprising:

an input for receiving source data that is represented by an ordered plurality of source symbols and the source data is recoverable from encoding symbols that are transmitted;

storage for at least a portion of a plurality of base blocks, wherein each base block comprises a representation of one or more source symbols of the ordered plurality of source symbols;

a logical map, stored in a machine-readable form or generatable according to logic instructions, mapping each of a plurality of source blocks to one or more base blocks, wherein the plurality of source blocks includes at least one pair of source blocks that have a characteristic that there is at least one base block that is enveloped by both source blocks of the pair and at least one base block for each source block of the pair that is enveloped by that source block and not by the other source block of the pair;

storage for encoded blocks;

one or more encoders that each encode source symbols of a source block to form a plurality of encoding symbols, with a given encoding symbol being independent of source symbol values from source blocks other than the source block it encodes source symbols of, such that a portion of the source data that is represented by a union of the pair of source blocks is assured to be recoverable from a combination of a first set of encoding symbols generated from a first source block of the pair and a second set of encoding symbols generated from a second source block of the pair, wherein an amount of encoding symbols in the first set is less than an amount of source data in the first source block and an amount of encoding symbols in the second set is less than an amount of source data in the second source block; and

an output for outputting the encoding symbols for transmission to the receiver over the communications channel.

41. The encoder of claim 40, wherein a number of encoding symbols in the first set plus a number of encoding symbols in the second set is no greater than a number of source symbols in the portion of the source data that is represented by the union of the pair of source blocks, if the encoding symbols and the source symbols have a same size.

42. The encoder of claim 40, wherein a portion of the source data that is represented by the first source block of the pair is recoverable from a third set of encoding symbols generated from the first source block, wherein an amount of encoding symbols in the third set is no greater than the amount of source data in the first source block.

43. The encoder of claim 40, wherein a number of distinct encoding symbols that can be generated from each source block is independent of a size of the source block.

44. The encoder of claim 40, further comprising:  
an input for receiving receiver feedback representing results at a decoder that is receiving or has received encoding symbols; and

logic for adjusting one or more of membership of source symbols in base blocks, membership of base blocks in enveloping source blocks, number of source symbols per base block, number of symbols in a source block,

37

and/or number of encoding symbols generated from a source block, wherein the adjusting is done based on, at least in part, the receiver feedback.

45. The encoder of claim 40, further comprising:  
an input for receiving data priority preference signals representing varying data priority preferences over the source data; and  
logic for adjusting one or more of membership of source symbols in base blocks, membership of base blocks in enveloping source blocks, number of source symbols per base block, number of symbols in a source block, and/or number of encoding symbols generated from a source block, wherein the adjusting is done based on, at least in part, the data priority preference signals.

46. The encoder of claim 40, wherein a number of source symbols in the base blocks enveloped by each source block is independent, as between two or more of the source blocks.

47. The encoder of claim 40, wherein source symbols identified to a base block are consecutive within the ordered plurality of source symbols.

38

48. The encoder of claim 40, wherein source symbols identified to the base blocks enveloped by a source block are consecutive within the ordered plurality of source symbols.

49. The encoder of claim 40, wherein a number of distinct encoding symbols that can be generated for a source block is independent of a number of encoding symbols that can be generated for other source blocks.

50. The encoder of claim 40, wherein a number of distinct encoding symbols generated for a given source block is independent of a number of source symbols in base blocks enveloped by the given source block.

51. The encoder of claim 40, further comprising:  
storage for a set of coefficients selected from a finite field for each of a plurality of the encoding symbols, wherein the one or more encoders further comprise

logic for generating the encoding symbol as a combination of source symbols of one or more base blocks enveloped by a single source block, wherein the combination is defined, in part, by the set of coefficients.

\* \* \* \* \*