Reliable Multicast Transport
Internet Draft

E. Stauffer
Broadcom
B. Shen
Broadcom
S. Chakraborty
Broadcom
D Tujkovic
Broadcom
Jing Huang
Broadcom
K. Rath
Broadcom

Intended status: Standards Track
Expires: November 2012

May 13, 2012

Supercharged Codes
draft-stauffer-rmt-bb-fec-supercharged-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes a fully-specified FEC scheme for the Supercharged forward error correction code. Supercharged codes are designed for use on the erasure channel. Coding for the erasure channel commonly arises for data transmission over the internet, where lower layers either successfully deliver packets or fail to deliver them. Coding is required to insure that data is not lost, even if packets are lost at the lower layers. Error free reception is important for multimedia applications, such as streaming, where it may not be possible to correct an error in time by any other means. Coding insures that lost packets can be recovered.

Table of Contents

1. Introduction

This document describes a fully-specified FEC scheme for the Supercharged forward error correction code. The Supercharged code is designed for the erasure channel with performance very close to the ideal Maximum Distance Separable(MDS) code and with very low complexity. Section 2 describes the architecture of the code and defines the generator matrices used by the code. Section 3 describes how to construct FEC packets. Section 4 discusses code parameter selection for a particular usage context. Section 5 defines the protocol information elements. Section 6 considers security. Section 7 considers IANA.

2. Supercharged Code

2.1.1. Definitions

ceil(a): rounds a to the nearest integer towards infinity

floor(a): rounds a to the nearest integer towards minus infinity

min(a,b): returns the minimum of a and b

max(a,b): returns the maximum of a and b

a % b: is a modulo b

a + b: is a plus b

a * b: is a multiplied by b.

a ^ b: the bitwise XOR of a and b

a ^^ b: raises a to the b power

I_a: the a x a identity matrix

zeros(a,b): the a x b zero matrix

2.2. Overview

Figure 1 shows a general block diagram of the supercharged code. It consists of a network of codes including block codes, repetition codes, and parallel filter codes. Block code 1 consists of a Vandermonde matrix in GF(256), a non-systematic Reed Solomon code. Block code 2 and 3 consist of binary block codes.

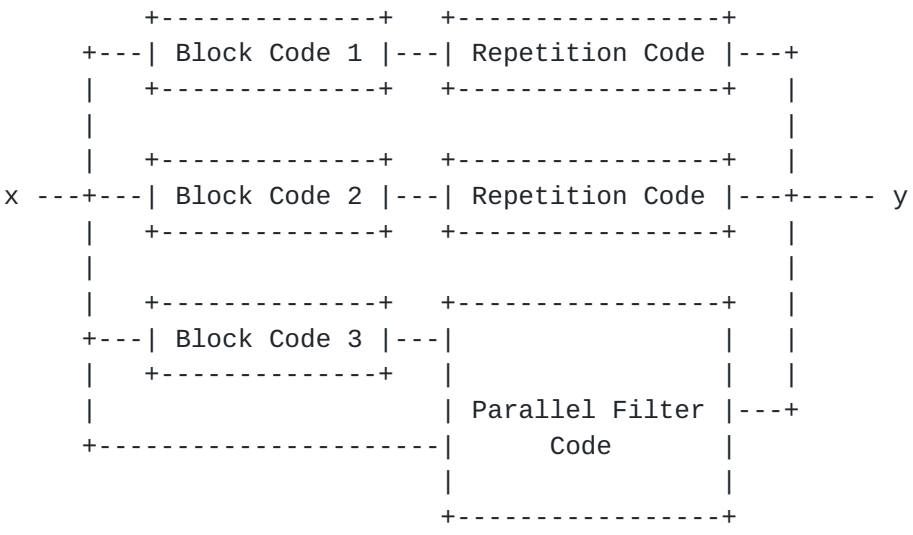

Figure 1 Block Diagram of the SC Code

The parallel filter code of Figure 1 is detailed in Figure 2. It consists of interleavers, tailbiting FIR filters, and a multiplexer to select the output of the filters.

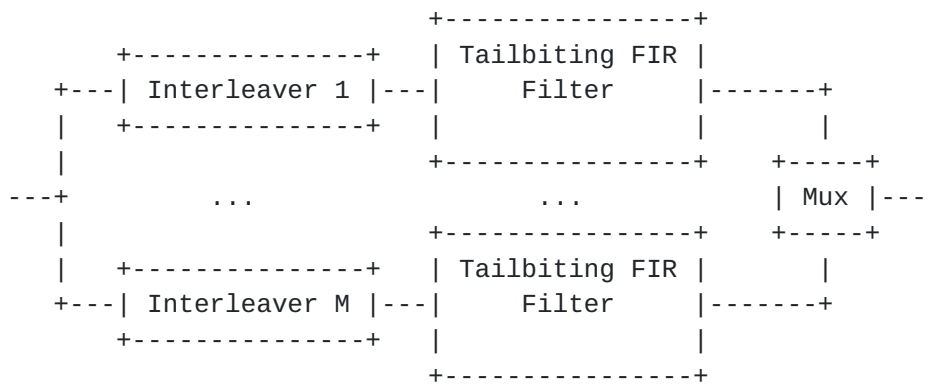

Figure 2 An example parallel filter code showing individual data interleavers and tailbiting FIR filters as coding components.

An example of one of the tailbiting FIR filters is illustrated in Figure 3, where the state of the filter is initialized with the final state to make it tailbiting.

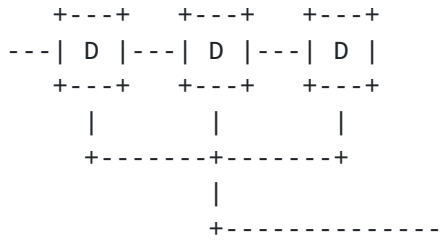

Figure 3 An example 3 tap FIR filter that can be used for the tailbiting FIR filter coding component. An XOR operation is applied at the output of the delay elements to produce the final output.

Optionally, if the number of transmit symbols N is signaled to be limited such that N<=256, then the code can achieve ideal performance by utilizing a Reed Solomon code.

2.3. Matrix Representation

Since supercharged codes are linear, an output codeword can be expressed as a matrix multiplied by an input vector. Given Kx1 encoding state vector x, consisting of binary transmit symbols, the output Nx1 codeword, y, can be written as

$$y = (T*[I_K; B_3] + R_1*B_1 + R_2*B_2)*x \quad (1)$$

where T is the N x (K+Num_B_3) generator matrix for the FIR structure, B_1 is the Num_V_RS x K generator matrix for the first block code, B_2 is the Num_B_2 x K generator for the second block code, B_3 is the Num_B_3 x K generator matrix of the third block code, and R_1 is a N x Num_V_RS stack and R_2 is a N x Num_B_2 stack of identity matrices which facilitates repetition. For example, matrix R_1 would consist of floor(N/Num_V_RS) copies of the identity matrix stacked vertically, with a fractional identity matrix below consisting of N mod Num_V_RS rows. The "+" operator indicates the bitwise XOR operation. For convenience, denote the generator matrix P = (T*[I_K; B_3] + R_1*B_1 + R_2*B_2), such that y=Px.

2.4. Systematic Encoding

Supercharged codes are not inherently systematic codes. Non-systematic codes are commonly transformed into an effective systematic code by pre-processing the input data before using it as the input to the encoder, y=Px. The encoder input is calculated by decoding the desired input data and running the decoder to determine the encoder input vector x. Let matrix P_enc be the KxK generating matrix corresponding to the first K elements of y, the encoder input x can be computed using the following

$$x = P\_enc^{-1} * d.$$

Now, x can be used to encode using equation (1) to generate y. The first K elements of vector y will be equal to d.

2.5. Erasure Channel

After encoding, the N transmit symbols of codeword vector y are transmitted on the channel. Some of these transmit symbols are erased by the channel. Suppose that the Nxr matrix E represents the erasure pattern of the channel in that it selects out the r received transmit symbols y_r from the transmitted symbols y. If the ith received symbol is the jth transmit symbol, then E(i,j)=1. This results in

$$y\_r = E*y.$$

At the decoder, the effective generator matrix at the receiver is P_r = E*P.

2.6. Decoding

Decoding is the process of determining x given y_r and P_r. Decoding can be implemented in several different ways, but each are equivalent

to solving the least squares problem x = (P_r^^T*P_r)^^-1 * P_r^^T * y_r. Modern sparse matrix factorization techniques can take advantage of the sparse structure imposed by the parallel filter structure if (1) is rewritten in the following equivalent form

$$z = Gw, \qquad (2)$$

with augmented generator matrix G defined as

$$G = [\ [B_3; B_2; B_1]\ I_L;\ T\ R_2\ R_1]$$

and where the augmented output vector z=[zeros(L,1); y], the augmented input vector w=[x; B_3*x; B_2*x; B_1*x], and where L= Num_V_RS+Num_B_2+Num_B_3. The bottom L elements of vector w contain the outputs, before repetition, of the block codes. These L values are appended to vector x to form the augmented input vector w. The first L rows of G implement the block code and XOR the block code output with itself to generate the L zeros at the top of the z vector. The subsequent N rows of G implement the FIR structure and XOR the output with the output of the block codes.

This problem can be efficiently solved using direct sparse matrix factorization techniques described in [3-8]. It is RECOMMENDED that the Dulmage-Mendelsohn based solver in chapter 8 of [5] be used with addition, multiplication, and division updated to support a finite field. This algorithm utilizes pivoting based on node degrees in the equivalent graph to minimize fill-in. The solution is completed by performing forward and backward substitutions. Iterative solvers are also possible.

Once the encoder state vector x, or equivalently the augmented encoder state vector w, has been determined, the task remains to determine the data vector d. For any elements of d that are missing, then can be recovered by using appropriate rows of (1) or (2).

2.7. Matrix P Construction

2.7.1. Function Prototypes

The following functions are utilized to construction the Supercharged code.

[K_eff, Num_V_RS, Num_B_2, Num_B_3] = SC_Parameters(K, N)

K_eff=SC_K_table(K)

```
b=RNG(a)

a=RNG_2(a,b)

[permutation,the_seed]= Generate_Permutation(a,b)

G_V_RS = RS_gen(K,N)

[filter_data, filter_N]=SC_filter_data(z)

b=GF_exp(a)

C=GF_Multiply(A,B)
```

2.7.2. Parallel Filter Code T Construction

The parallel filter code matrix T can be generated using the following pseudo code. The code generates multiple random interleavers and selects which output of which interleaver depending on the SID, where the SID is definded in section 3. Note that at the receiver, only filter outputs corresponding to the received SID's are required. The following code generates filter outputs for SIDs 0 to N-1. Determination of the filter output is a function of the SID only, not any other filter output, making it simple to generate only the filter outputs needed at encoding or decoding. The Generate_Permutation function is defined in section 2.7.9. , the SC_filter_data function is defined in section 2.7.12. , and the RNG function is defined in section 2.7.8.

```
    seed1 = 758492

    seed2 = ( ((K_eff*874) % (2^32)) ^ (seed1) )

    seed3 = 23091

    base_permutation = Generate_Permutation(K_eff+Num_B_3,seed2)

    filter_data = SC_filter_data(K_eff+NUM_B_3)


    T = zeros(N,K_eff+NUM_B_3)

    for SID=0:N-1
```

```
          %Determine which filter to select

          rn1 = RNG(15*loop+2*seed3)

          index = 0

          while(rn1>(filter_data[index+1]))

               index = index+1

          end


          %Determine which interleaver to select

          rn2 = RNG(2*K_eff+3*SID)

          interleaver_number = ( (rn2) % (K_eff+NUM_B_3) )


          %Determine which part of the interleaver to select

          rn3 = RNG(98573+2*SID+rn3)

          interleaver_part = ((rn3) % (K_eff+NUM_B_3))


          for tap_loop=0:tdeg-1

              filter_tap = (tap_loop+interleaver_part) %
   (K_eff+NUM_B_3)

              tap_location = (base_permutation[filter_tap] +
   base_permutation[interleaver_number]) % (K_eff+NUM_B_3)

              T[Num_V_RS+Num_Rep+SID,tap_location] = 1

          end

      end
```

2.7.3. Repetition Code R Construction

The repetition code matrix R_1 and R_2 can be constructed via the following pseudo code. Note that at the receiver, only filter outputs corresponding to the received SID's are required. The following code generates filter outputs for SIDs 0 to N-1 for R_1.

```
R_1 = zeros(N,Num_V_RS)

 for SID = 0:N-1

     for k = 0:Num_V_RS-1

         if( ((SID-k) % (Num_V_RS)) == 0 )

             R_1[SID,k] = 1

         end

     end

 end
```

The following code generates filter outputs for SIDs 0 to N-1 for R_2.

```
R_2 = zeros(N, Num_B_2)

for SID = 0:N-1

    for k = 0: Num_B_2-1

        if( ((SID-k) % (Num_B_2)) == 0 )

            R_2[SID,k] = 1

        end

    end

end
```

2.7.4. Block Code B_1 Construction

The Vandermonde matrix of block code B_1 can be constructed via the following pseudo code. The GF_exp function is defined in section 2.7.13.

```
 B_1 = zeros(Num_V_RS,K_eff)

for i = 0:Num_V_RS-1

    for k = 0:K_eff-1

        B_1[i+1,k+1] = GF_exp( ((i+1)*k) % (2^^8-1) )

    end

end
```

2.7.5. Block Code B_2 and B_3 Construction

The block code B_2 and B_3 can be constructed jointly via the following pseudo code, where B_23=[B_3; B_2].

```
B_23 = zeros(Num_B_2 + Num_B_3,K_eff)

for i = 0:K_eff-1

    for k = 0: Num_B_2 + Num_B_3 - 1

        if( ( (k-i) % (Num_B_2 + Num_B_3) ) == 0)

            B_23[k,i] = 1

        end

    end

end
```

```
      m=1

      for i = 0:K_eff-1

          for k = 0: Num_B_2 + Num_B_3 - 1

              if( ( (k-i-2*floor(m/( Num_B_2 + Num_B_3))) % (Num_B_2 +
  Num_B_3) ) == 0)

                  B_23[k,i] = 1

              end

              m = m+1

          end

      end


      m=2

      for i = 0:K_eff-1

          for k = 0: Num_B_2 + Num_B_3 - 1

              if( ( (k-i-3*floor(m/( Num_B_2 + Num_B_3))) % (Num_B_2 +
  Num_B_3) ) == 0)

                  B_23[k,i] = 1

              end

              m = m+1

          end

      end
```

2.7.6. SC_Parameters

The following pseudo code determines a set of parameters needed for matrix construction. The SC_K_table is defined in section 2.7.7.

```
function [K_eff, Num_V_RS, Num_B_2, Num_B_3] = SC_Parameters(K, N)

     K_eff = SC_K_table(K)

    Num_V_RS = 16 + floor(K_eff/10000)

    Num_B = floor(K_eff^^(0.62)) + 3

    if( K_eff >= 17376 )

        Num_B = ceil( K_eff*0.0152 + 163 )

    end

    Num_B_3 = ceil(0.75*( Num_B ))

    Num_B_2 = Num_B  Num_B_3
```

2.7.7. K Table

The function K_eff=SC_K_table(K) is implemented based on the following table, by returning the smallest K_eff such that K_eff>=K.

```
10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,
56,57,58,59,60,61,62,63,64,65,66,67,69,70,71,72,73,74,75,76,77,78,79,
80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,1
02,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,11
9,120,121,122,123,124,125,126,127,128,129,130,131,133,134,135,136,137
,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,
155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,1
72,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,18
9,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206
,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,
```

224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,2
41,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,25
8,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275
,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,
293,294,295,296,297,298,299,300,302,303,304,305,306,307,308,309,310,3
11,312,314,315,316,320,321,324,328,329,335,337,338,340,341,344,347,34
9,352,355,357,358,360,362,364,366,368,372,377,380,381,382,384,385,388
,389,393,394,395,397,399,405,408,409,410,411,416,418,424,426,428,431,
432,434,438,443,447,448,451,452,453,457,460,465,466,467,469,473,476,4
77,478,482,483,484,485,486,490,491,492,493,494,496,497,498,500,501,50
2,503,504,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520
,521,522,524,526,527,528,529,530,532,533,534,535,536,537,539,541,542,
543,545,546,549,551,552,553,554,555,557,558,559,561,562,563,564,566,5
69,571,572,573,574,576,577,578,579,580,582,583,585,586,587,588,589,59
0,592,593,594,597,598,599,600,602,603,606,607,608,609,610,612,614,615
,616,617,619,620,622,625,626,627,628,629,630,631,633,635,636,637,638,
640,643,645,648,650,652,653,654,655,656,659,660,661,662,664,666,667,6
68,669,672,673,674,675,677,687,688,691,692,693,694,695,696,698,699,70
0,701,703,710,711,712,715,716,717,718,726,727,730,731,734,736,737,741
,744,747,748,751,752,753,757,759,760,762,764,766,769,771,772,773,774,
775,777,778,779,786,788,790,792,793,794,795,797,798,799,800,801,802,8
04,805,810,811,812,813,815,820,821,822,823,825,827,829,830,831,834,83
5,837,838,839,840,843,844,845,846,848,849,851,852,853,854,857,858,860
,863,864,866,868,869,870,875,877,879,883,886,887,890,891,894,897,898,
899,900,902,903,904,905,906,907,909,912,913,914,917,922,926,927,928,9
31,934,938,940,942,944,945,948,950,953,954,960,961,963,967,968,970,97
1,972,974,977,979,980,981,985,987,989,990,995,996,1000,1002,1003,1005
,1006,1007,1009,1010,1015,1020,1021,1022,1024,1025,1027,1032,1033,103
4,1035,1037,1041,1042,1043,1046,1048,1050,1051,1054,1056,1057,1059,10
60,1062,1065,1069,1070,1071,1074,1076,1078,1079,1082,1083,1085,1086,1
087,1088,1089,1095,1098,1099,1106,1110,1111,1118,1120,1123,1124,1125,
1131,1132,1134,1136,1139,1140,1142,1144,1150,1152,1157,1161,1162,1165
,1169,1173,1175,1176,1179,1181,1182,1183,1194,1200,1201,1204,1205,120
6,1208,1209,1212,1213,1214,1218,1219,1220,1222,1225,1227,1228,1229,12
32,1236,1238,1240,1242,1243,1245,1248,1250,1252,1253,1255,1258,1261,1
269,1273,1278,1279,1280,1283,1284,1292,1293,1302,1303,1306,1310,1311,
1315,1318,1319,1321,1325,1330,1331,1342,1343,1347,1348,1352,1357,1359
,1361,1365,1374,1380,1382,1384,1388,1389,1390,1391,1392,1395,1397,140
3,1404,1407,1413,1417,1418,1420,1425,1429,1431,1435,1436,1437,1447,14
50,1461,1462,1464,1473,1474,1475,1477,1485,1490,1494,1496,1497,1502,1
503,1507,1513,1514,1516,1521,1522,1526,1530,1534,1539,1541,1549,1552,
1554,1555,1561,1564,1569,1572,1579,1585,1586,1590,1591,1593,1595,1596
,1597,1598,1600,1604,1608,1610,1611,1612,1616,1617,1624,1631,1633,163
6,1641,1646,1649,1650,1658,1660,1665,1667,1671,1673,1679,1683,1689,16
92,1696,1698,1703,1705,1707,1708,1713,1716,1722,1728,1733,1734,1739,1
740,1742,1744,1745,1756,1759,1760,1764,1768,1771,1776,1777,1780,1782,
1787,1800,1807,1814,1824,1826,1827,1842,1844,1854,1857,1863,1867,1873

,1874,1878,1881,1883,1887,1889,1890,1891,1892,1894,1896,1903,1905,190
6,1910,1919,1924,1926,1931,1933,1943,1944,1948,1952,1954,1967,1971,19
73,1976,1979,1985,1986,1987,1989,1992,1994,1995,1998,2000,2005,2006,2
018,2019,2030,2040,2043,2048,2054,2055,2057,2061,2070,2071,2074,2077,
2082,2084,2087,2089,2093,2096,2098,2103,2104,2107,2111,2120,2122,2125
,2128,2138,2150,2152,2155,2160,2175,2177,2182,2189,2195,2200,2201,220
3,2217,2219,2225,2226,2231,2234,2235,2236,2237,2245,2247,2274,2276,22
78,2280,2282,2283,2286,2292,2303,2304,2306,2310,2315,2316,2319,2320,2
321,2330,2333,2336,2339,2343,2344,2345,2351,2367,2368,2371,2374,2382,
2389,2392,2395,2396,2400,2402,2407,2410,2412,2416,2421,2422,2434,2442
,2446,2447,2462,2473,2477,2478,2481,2486,2490,2492,2495,2502,2505,250
7,2509,2512,2513,2522,2525,2527,2528,2536,2543,2549,2556,2559,2561,25
63,2565,2583,2587,2590,2592,2596,2598,2601,2603,2604,2606,2617,2622,2
625,2626,2636,2638,2640,2643,2654,2660,2668,2673,2677,2679,2688,2695,
2699,2701,2713,2714,2723,2737,2741,2747,2753,2762,2764,2769,2772,2775
,2776,2785,2796,2802,2805,2808,2826,2828,2830,2831,2834,2836,2853,287
5,2877,2878,2884,2906,2938,2945,2948,2950,2961,2964,2966,2968,2979,29
80,2985,2989,2998,3008,3011,3015,3018,3022,3027,3048,3049,3051,3053,3
056,3062,3071,3075,3080,3093,3094,3095,3097,3101,3107,3109,3119,3122,
3128,3149,3150,3151,3158,3166,3167,3173,3178,3180,3181,3182,3186,3190
,3195,3200,3201,3203,3204,3205,3208,3216,3217,3223,3224,3232,3236,324
0,3248,3251,3253,3269,3276,3278,3279,3286,3292,3299,3306,3309,3336,33
40,3342,3344,3351,3352,3356,3357,3371,3375,3380,3387,3396,3404,3407,3
410,3423,3430,3445,3451,3463,3466,3471,3478,3479,3502,3513,3520,3528,
3531,3534,3539,3540,3546,3551,3565,3577,3579,3603,3606,3608,3612,3614
,3616,3620,3647,3650,3653,3658,3664,3677,3682,3686,3694,3697,3705,370
7,3724,3728,3744,3749,3751,3754,3761,3765,3776,3778,3781,3792,3797,37
99,3801,3834,3840,3841,3848,3861,3863,3883,3901,3903,3919,3924,3941,3
943,3960,3965,3970,3971,3989,3992,4007,4013,4015,4037,4039,4045,4050,
4055,4069,4072,4073,4091,4096,4106,4112,4124,4129,4133,4140,4146,4156
,4165,4188,4207,4209,4210,4215,4221,4236,4237,4247,4252,4253,4257,426
1,4266,4270,4318,4330,4341,4346,4359,4363,4365,4366,4388,4415,4418,44
36,4438,4453,4468,4474,4477,4503,4512,4513,4519,4522,4538,4548,4567,4
575,4576,4577,4583,4590,4621,4639,4651,4659,4681,4693,4698,4700,4702,
4729,4731,4739,4741,4742,4748,4749,4758,4764,4765,4771,4772,4780,4785
,4803,4804,4838,4840,4843,4868,4871,4878,4885,4898,4901,4918,4924,493
3,4939,4954,4959,4979,4982,4988,4991,4999,5000,5008,5021,5023,5030,50
39,5060,5062,5063,5096,5116,5137,5143,5145,5162,5163,5167,5172,5186,5
218,5225,5238,5240,5252,5260,5279,5285,5295,5301,5310,5314,5317,5331,
5332,5334,5348,5353,5354,5390,5391,5392,5405,5407,5432,5449,5451,5453
,5460,5464,5466,5471,5473,5477,5492,5506,5508,5537,5540,5543,5554,556
1,5566,5570,5576,5579,5587,5616,5637,5672,5674,5676,5684,5694,5716,57
32,5774,5792,5798,5800,5808,5823,5838,5844,5863,5896,5897,5899,5900,5
916,5921,5930,5960,5975,6039,6055,6057,6059,6067,6068,6078,6092,6099,
6102,6107,6136,6151,6169,6189,6191,6218,6233,6249,6271,6274,6296,6318
,6352,6363,6376,6407,6430,6435,6441,6463,6486,6491,6502,6512,6518,652
0,6534,6542,6549,6553,6589,6590,6593,6599,6614,6625,6634,6643,6655,66

70,6680,6684,6691,6692,6701,6708,6711,6724,6730,6732,6752,6799,6803,6
809,6812,6834,6849,6855,6877,6878,6879,6899,6907,6919,6936,6945,6946,
6954,6955,6956,6958,6981,7000,7011,7030,7032,7033,7108,7111,7127,7164
,7171,7175,7179,7181,7185,7225,7226,7281,7288,7295,7307,7325,7359,736
0,7390,7392,7411,7476,7520,7535,7548,7552,7558,7567,7589,7596,7616,76
45,7675,7679,7714,7726,7747,7770,7780,7785,7805,7818,7855,7870,7883,7
923,7935,7936,7953,7974,7999,8028,8030,8069,8074,8093,8104,8111,8122,
8150,8154,8172,8173,8189,8192,8193,8194,8223,8236,8290,8304,8377,8425
,8438,8439,8464,8481,8492,8521,8556,8559,8575,8582,8595,8602,8606,862
4,8628,8648,8654,8666,8672,8689,8738,8739,8744,8775,8787,8837,8841,88
42,8860,8928,8929,8970,8977,8993,9009,9019,9020,9029,9041,9051,9087,9
111,9151,9195,9208,9298,9303,9327,9344,9352,9360,9364,9388,9400,9402,
9446,9448,9449,9461,9462,9470,9485,9497,9512,9539,9546,9560,9572,9601
,9612,9642,9649,9653,9677,9689,9692,9704,9708,9758,9765,9794,9813,986
0,9916,9922,9927,9949,9971,9978,9981,9986,9987,10017,10040,10065,1007
3,10084,10097,10105,10120,10124,10134,10166,10187,10197,10202,10204,1
0241,10242,10279,10308,10324,10336,10351,10361,10458,10460,10567,1064
3,10676,10705,10712,10717,10759,10786,10787,10857,10883,10899,10911,1
0933,10944,10958,10963,11011,11015,11024,11036,11039,11049,11060,1111
9,11130,11146,11172,11203,11210,11216,11219,11230,11245,11316,11358,1
1371,11376,11423,11475,11534,11590,11649,11653,11677,11686,11707,1171
1,11740,11748,11751,11780,11823,11829,11843,11890,11896,11919,11947,1
1956,11976,12026,12037,12045,12072,12087,12108,12119,12154,12160,1220
8,12215,12216,12228,12229,12235,12247,12294,12333,12400,12437,12455,1
2458,12460,12469,12471,12510,12528,12567,12569,12593,12685,12694,1270
4,12721,12726,12754,12790,12817,12857,12914,12928,12936,12956,13002,1
3012,13026,13030,13035,13038,13057,13067,13082,13114,13143,13159,1319
3,13204,13214,13270,13278,13284,13326,13335,13417,13421,13423,13460,1
3479,13558,13607,13695,13696,13742,13764,13816,13827,13833,13837,1387
4,13879,13974,13987,14022,14100,14115,14140,14202,14272,14342,14350,1
4370,14376,14385,14393,14408,14409,14415,14417,14442,14486,14509,1456
0,14565,14713,14729,14743,14755,14798,14862,14874,14913,14934,14990,1
5007,15011,15120,15170,15194,15217,15227,15235,15285,15314,15321,1532
5,15332,15438,15499,15573,15611,15651,15668,15732,15735,15741,15757,1
5780,15808,15813,15847,15870,15941,15953,15977,16002,16017,16060,1610
8,16161,16286,16287,16304,16336,16374,16377,16384,16414,16505,16563,1
6623,16665,16670,16674,16689,16691,16710,16727,16743,16794,16828,1685
1,16900,16974,17005,17024,17029,17038,17039,17051,17086,17098,17148,1
7151,17195,17206,17266,17316,17323,17326,17331,17357,17376,17466,1748
9,17531,17559,17642,17681,17791,17868,17926,17929,17988,17991,18009,1
8026,18027,18056,18116,18168,18232,18307,18309,18438,18503,18504,1851
1,18590,18628,18629,18630,18636,18647,18672,18691,18694,18719,18909,1
8988,19023,19036,19096,19126,19132,19139,19193,19204,19210,19277,1930
4,19314,19325,19539,19544,19547,19631,19632,19635,19675,19700,19705,1
9740,19748,19921,19939,19951,19972,19985,20042,20052,20133,20141,2015
2,20173,20230,20245,20269,20287,20335,20355,20396,20407,20455,20501,2
0564,20580,20583,20664,20683,20710,20768,20776,20778,20789,20794,2098

8,21058,21087,21141,21143,21151,21186,21199,21216,21224,21385,21412,2
1468,21475,21478,21479,21486,21487,21515,21569,21616,21629,21673,2170
2,21729,21737,21747,21852,21927,21969,22060,22062,22068,22073,22114,2
2131,22244,22301,22320,22366,22433,22450,22482,22490,22498,22536,2272
7,22787,22947,22994,23010,23026,23063,23084,23135,23158,23180,23252,2
3392,23457,23491,23500,23568,23607,23721,23730,23787,23935,23971,2399
1,24023,24185,24215,24232,24398,24406,24476,24548,24550,24555,24562,2
4566,24591,24592,24616,24633,24673,24721,24735,24743,24761,24832,2489
1,24967,24976,25062,25080,25230,25391,25407,25433,25463,25493,25543,2
5613,25668,25756,25919,26022,26048,26050,26092,26291,26297,26329,2634
2,26371,26535,26566,26582,26676,26741,26838,26908,26910,26973,26984,2
7111,27119,27163,27256,27296,27353,27392,27428,27492,27594,27644,2766
6,27682,27771,27885,27895,27959,27987,28088,28116,28134,28137,28248,2
8263,28365,28466,28548,28549,28787,28816,28845,28966,29002,29042,2905
4,29072,29127,29138,29265,29326,29345,29434,29481,29487,29500,29588,2
9731,29816,29827,29868,29905,29964,30037,30097,30153,30169,30280,3034
6,30405,30433,30461,30493,30513,30550,30583,30646,30654,30909,30915,3
0921,30930,30974,30997,31052,31056,31142,31199,31283,31285,31303,3150
5,31578,31605,31948,31957,31997,32124,32139,32142,32272,32403,32555,3
2601,32630,32631,32648,32699,32768,32807,32849,32912,32932,32961,3296
5,33129,33171,33200,33282,33334,33623,34258,34302,34654,34708,35024,3
5031,35388,35395,35462,35488,35586,35600,35747,35750,35774,35802,3607
1,36112,36189,36252,36254,36294,36328,36357,36448,36476,36477,36479,3
6485,36637,36749,36849,36874,36894,37170,37185,37187,37227,37612,3769
5,37701,37767,37793,37805,37815,37826,37906,37992,38008,38010,38046,3
8080,38130,38236,38385,38763,38787,39166,39176,39201,39237,39288,3939
8,39482,39643,39786,39831,39960,39980,40089,40105,40140,40152,40192,4
0220,40274,40293,40303,40398,40549,40604,40625,40666,40690,40816,4084
3,40847,40894,40896,40962,40969,41003,41087,41107,41132,41216,41226,4
1265,41314,41321,41357,41367,41539,41576,41641,41717,41820,42033,4206
7,42172,42490,42662,42795,42813,42916,43339,43351,43388,43482,43498,4
3691,43840,43905,43924,43932,44033,44129,44279,44821,44883,44945,4495
1,45097,45162,45359,45389,45557,45582,45638,45813,45830,45919,45960,4
6038,46086,46104,46187,46281,46428,46463,46481,46574,47047,47324,4741
8,47523,47717,48007,48264,48334,48489,48501,48702,48788,48976,48994,4
9504,49550,49703,49711,49978,49995,50006,50338,50511,50799,50946,5094
7,50951,50980,51017,51150,51244,51530,51616,51977,52007,52062,52364,5
2441,52586,52598,52768,52883,52978,53047,53064,53114,53127,54024,5454
6,54578,54735,54803,55123,55289,55510,55661,55744,55843,55885,55921,5
6297,56403,56696,57113,57424,57614,57779,58294,58326,58721,58908,5934
6,59541,59651,59882,60076,60164,60250,60618,60799,61144,61208,61217,6
1617

2.7.8. Random Number Generator

The SC code utilizes two random number generators. The first uses the second. The first is described by the following pseudo code:

```
function b=RNG(a)

for i = 0:7

    a = RNG_2( a, ( (a) % (89) ) + 1 )

    b = (b) % (a)

end
```

The second random number generator uses a selectable set of feedback taps. The second is described by the following pseudo code:

```
function a=RNG_2(a,b)

tap_list=[32, 31, 30, 10

32, 31, 29, 1

32, 31, 26, 18

32, 31, 26, 9

32, 31, 26, 7

32, 31, 23, 10

32, 31, 22, 17

32, 31, 21, 16

32, 31, 21, 5

32, 31, 18, 10

32, 31, 16, 2

32, 31, 15, 10

32, 31, 14, 4
```

32, 31, 13, 8

32, 31, 9, 7

32, 31, 5, 4

32, 30, 29, 23

32, 30, 29, 20

32, 30, 29, 16

32, 30, 29, 15

32, 30, 27, 24

32, 30, 27, 21

32, 30, 27, 12

32, 30, 27, 8

32, 30, 26, 25

32, 30, 26, 13

32, 30, 25, 16

32, 30, 23, 16

32, 30, 23, 14

32, 30, 23, 4

32, 30, 21, 14

32, 30, 19, 8

32, 30, 19, 4

32, 30, 17, 3

32, 30, 15, 6

32, 30, 11, 8

32, 30, 11, 5

32, 30, 8, 3

32, 30, 7, 4

32, 29, 28, 19

32, 29, 27, 23

32, 29, 27, 21

32, 29, 27, 6

32, 29, 26, 6

32, 29, 25, 6

32, 29, 22, 18

32, 29, 19, 16

32, 29, 17, 15

32, 29, 15, 8

32, 29, 6, 5

32, 29, 6, 4

32, 28, 25, 15

32, 28, 25, 11

32, 28, 25, 6

32, 28, 23, 6

32, 28, 15, 13

32, 28, 9, 7

32, 27, 26, 14

32, 27, 25, 20

32, 27, 25, 19

32, 27, 25, 17

32, 27, 25, 7

32, 27, 25, 5

32, 27, 23, 6

32, 27, 21, 6

32, 27, 20, 18

32, 27, 18, 14

32, 27, 15, 14

32, 27, 14, 12

32, 27, 14, 9

32, 27, 8, 6

32, 26, 25, 10

32, 26, 23, 12

32, 26, 22, 7

32, 26, 20, 11

32, 26, 19, 9

32, 26, 19, 7

32, 26, 18, 13

32, 26, 15, 7

32, 25, 24, 7

32, 25, 22, 15

32, 25, 17, 7

32, 25, 14, 13

32, 24, 22, 13

32, 23, 21, 16

```
   32, 23, 18, 14

   32, 21, 20, 19

   32, 20, 17, 15

   32, 19, 18, 13]

   taps[0]=tap_list[b,0]

   taps[1]=tap_list[b,1]

   taps[2]=tap_list[b,2]

   taps[3]=tap_list[b,3]

   feedback=2.^^(32-taps[0]) + 2.^^(32-taps[1]) + 2.^^(32-taps[2]) +
   2.^^(32-taps[3])

   if( (a) & (1) )

       a = (a) ^ (feedback)

       a = (a) >> (1)

       a = (2^31) || (a)

   else

       a = (a) >> (1)

   end
```

2.7.9. Random Permutation

The SC code utilizes a random permutation of length K to facilitate the construction of the random interleavers needed for the parallel filter codes. The random permutation is given by the following pseduocode. The RNG_2 function is defined in section 2.7.8.

```
   function [permutation,the_seed]= Generate_Permutation(a,b)

   for i=0:a-1

      permutation[i] = i + 1
```

```
   end


   for i=0:a-1
       c = RNG_2(b,1)
       b = ( (c) % (a-(i-1)) ) + i - 1
       d = permutation[i]
       permutation[i] = permutation[b]
       permutation[b] = d
   end
```

2.7.10. RS Generator

A Reed Solomon code is utilized in the construction of the SC code. Its construction is described by the following pseudo code. The GF_exp and the GF_Multiply functions are defined in section 2.7.13.

```
   function G_V_RS = RS_gen(K,N)
   Gt=zeros[N,K]
   for i=0:N-1
       for k=0:K-1
           a = ((i+1)*k) % (2^^8-1)
           Gt[i,k]=GF_exp(a)
       end
   end


   G1=Gt[1:K,1:K]
   G2=Gt[K+1:N,1:K]
```

```
G_V_RS = GF_Multiply(G2,G1^^-1)
```

GF_Multiply implementes G2*G1_inv where the multiplication and addition are performend in the GF field. The matrix inverse G1^^-1 can be easily implemented using Gaussian Elimination for the small matrix G1.

2.7.11. RS Code

If the number of transmit symbols N is optionally limited to N<=256 and signaled using Max_N=0, then the following pseudo code is used to generate matrix P. The RS_gen function is defined in section 2.7.10.

```
Num_V_RS = N - K

B_1 = RS_gen(K,K+Num_V_RS)

P = [I[K]

     B_1 ]

Num_B = 0

K_eff = K
```

2.7.12. SC_Filter_Data

```
[filter_data, filter_N]=SC_filter_data(z)

Filter_data=[0,2147483648,2863311531,3221225472,3435973837,3579139413
,3681400539,3758096384,3817748708,3865470566,3904515724,3937053355,39
64585196,3988183918,4008636143,4026531840,4042322161,4056358002,40689
16386,4080218931,4090445044,4099741510,4108229587,4116010325,41231686
04,4129776246,4135894433,4141575607,4146864975,4151801719,4156419964,
4160749568,4164816772,4168644728,4172253945,4175662649,4178887099,418
1941841,4184839929,4187593114,4190211996,4192706170,4195084336,419735
4403,4199523578,4201598442,4203585013,4205488811,4207314902,420906795
0,4210752251,4212371771,4213930177,4215430865,4216876982,4218271451,4
219616993,4220916136,4222171240,4223384508,4224557996,4225693630,4226
793212,4227858432,4228890876,4229892034,4230863307,4231806012,4232721
393,4233610620,4234474799,4235314972,4236132128,4236927197,4237701065
```

```
,4238454568,4239188500,4239903613,4240600621,4241280205,4241943008,42
42589646,4243220702,4243836733,4244438269,4245025816,4245599856,42461
60849,4246709236,4247245437,4247769853,4248282869,4248784852,42492761
55,4249757114,4250228053,4250689283,4251141099,4251583788,4294967295]

filter_N=min(100,z)

Filter_data[Filter_N-1]=4294967295
```

2.7.13. GF(256) Operations

The SC code utilizes Galois field arithmetic in GF(256). The primitive polynomial is D^^8 + D^^4 + D^^3 + D^^2 + 1. The b=GF_exp(a) function raises the primitive element to the supplied power, a. The function C=GF_Multiply(A,B) multiplies two matrices in the Galois field.

3. FEC Packets

Encoded packets are constructed using a 4 byte FEC Payload IE followed by transmit symbols. The Source ID field (SID) of the FEC Payload IE identifies the Source ID of the first transmit symbol in the packet. Subsequent transmit symbols have sequential increasing SIDs. If the last transmit symbol of a packet contains source padding, these padding bytes may be excluded from the packet. Otherwise, packets must contain only whole transmit symbols.

It is RECOMMENDED that each packet include exactly one transmit symbol. Multiple transmit symbols per packet SHALL also be supported.

3.1. Segmentation

In order to encode large files within the working memory constraint, the source file may need to be segmented into transmit blocks and working blocks.

3.1.1. Transmit Blocks

Given a source file of size F bytes and a transmit symbol size of T bytes, the file can be divided into ceil(F/T) transmit symbols. A source transmit block is a collection of KL or KS of these transmit symbols. KL and KS may be different if the total number of source transmit blocks does not evenly divide the number of transmit symbols required to represent the file. The number of source transmit blocks with KL transmit symbols and the number of source transmit blocks

with KS transmit symbols are communicated to the decoder using parameters ZL and ZS, respectively. After encoding, a transmit block consists of a source transmit block and a repair transmit block.

The transmit blocks are ordered such that the first ZL transmit block are encoded from source transmit blocks of size KL transmit symbols. The remaining ZS transmit blocks are encoded from source transmit blocks are of size KS transmit symbols. The parameters KS and KL are related to ZS and ZL via KS = ceil((F/T-ZL) / (ZL+ZS)) and KL=KS+1.Working Blocks

In order to satisfy the working memory requirement, the transmit symbols can be further subdivided into working symbols of size TW bytes. A transmit symbol therefore consists of T/TW working symbols. A source working block is then a collection of working symbols selected such that an entire source working block can fit into the working memory. The ith source working block consists of the ith working symbol of transmit symbols of a source transmit block. Additionally, a source working block is to be sized such the size of the working symbols remain a multiple of the byte alignment factor, AL. The KL (or KS) transmit symbols of a source transmit block can be viewed as a collection of working symbols or equivalently as a collection of source working blocks.

After encoding, a working block consists of a source working block and a repair working block. The receiver attempts to decode on a subset of the source and repair working symbols in a working block. The working blocks are ordered in a packet such that the i th working symbol of TW bytes corresponds to the i th working block. The i th packet contains i working symbol for each of the working blocks.

3.1.2. Padding

In cases where effective number of transmit symbols used by the encoder and decoder, K_eff, is K_eff>K, then K_eff-K transmit symbols must be padded (with 0) to the data before encoding. These padded symbols do not need to be transmitted, as the decoder is aware that they are padding. (SIDs K to K_eff-1 MAY be transmitted, but it is RECOMMENDED that they are not.)

4. Parameter Selection

The code requires F, T, ZS, and TW. F is the total file size in

Bytes. T is the transmit symbol size in bytes, and it is RECOMMENDED

that it be equal to the packet payload size. The number of transmit blocks ZS with KS transmit symbols (the number of working blocks with KS working symbols) MUST be chosen such that KL<=K_max, where KL is computed in section 3.1.1. K_max is the maximum value in section 2.7.7.

The working symbols size in bytes, TW, MUST be chosen small enough such that K_L*TW is less than or equal to the working memory requirement. Additionally, TW MUST be chosen to be a multiple of the byte alignment factor AL and TW MUST be a divisor of T. The byte alignment, AL, is to be chosen based on the protocol and the typical machine architectures, a value of 4 (bytes) is RECOMMENDED.

5. Protocol IEs

This section describes IEs that are used by the FEC. All fields are big-endian.

The value of the FEC Encoding ID MUST be 7, as assigned by IANA (see Section 8).

5.1. FEC Payload IEs

The FEC payload ID is a 4-byte field defined as follows:

[0:7] TBN, (8 bits, unsigned integer): A non-negative integer identifier indicating the transmit block number.

[8:31] SID , (24 bits, unsigned integer): A non-negative integer identifier indicating the transmit symbols in the packet. SID 0 to K-1 indicate systematic symbols.

The FEC Payload ID is shown in Figure 4.

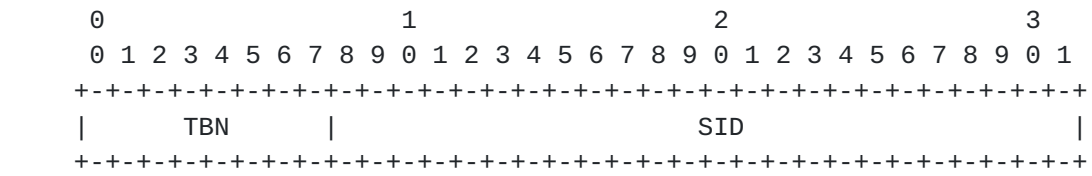

Figure 4 FEC Payload ID format

5.2. Common

The Common FEC Object Transmission Information elements used by this FEC Scheme are:

[0:39] Transfer Length (F), (40 bits, unsigned integer): A non-negative integer. This is the transfer length of the object in bytes.

[40:47] are reserved.

[48:63] Transmit Symbol Size (T), (16 bits, unsigned integer): A positive integer that is less than 2^^16. This is the size of a transmit symbol in units of bytes.

The encoded Common FEC Object Transmission Information format is shown in Figure 5.

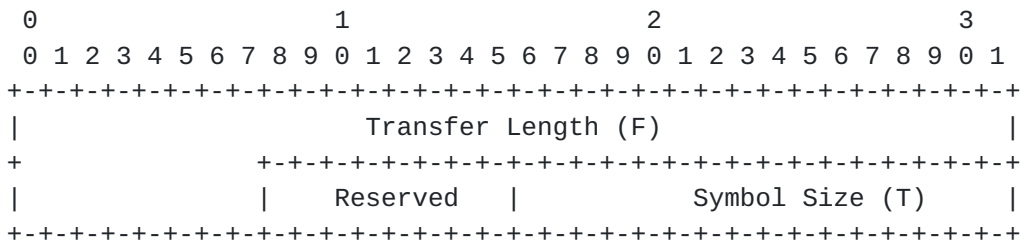

Figure 5 Encoded Common FEC IE for Supercharged FEC Scheme

5.3. Scheme Specific

The following parameters are carried in the Scheme-Specific FEC Object Transmission Information element for this FEC Scheme:

[0:7] ZL: The number of transmit blocks with KL working symbols (and the number of working blocks with KL working symbols). (8 bits, unsigned integer)

[8:15] ZS: The number of transmit blocks with KS working symbols (and the number ofworking blocks with KS working symbols). (8 bits, unsigned integer)

[16:30] TW: The working symbol size in bytes (15 bits, unsigned integer)

[31] Max_N: 0: OPTIONALLY Indicates that the maximum value of N satisfies N<=256. 1: Otherwise or if unknown (1 bit, boolean).

The encoded Specific FEC Object Transmission Information format is shown in Figure 5.

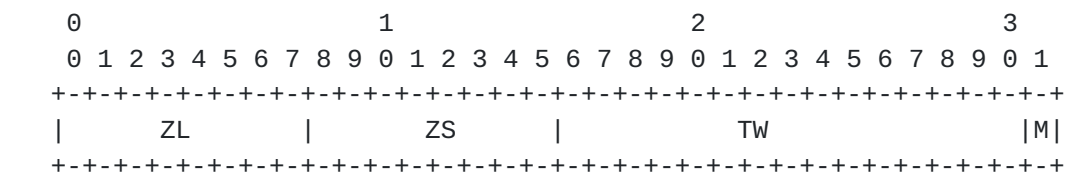

Figure 6 FEC Payload ID format

6. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

7. Security Considerations

Users could potentially be subject to a denial of service attack if a single erroneous packet is injected into the delivery stream. Therefore, it is RECOMMENDED that source authentication and integrity checking are applied to the file or data object before delivering decoded data to applications. The hashing methodology of SHA-256 is an example [2].

8. IANA Considerations

Values of FEC Encoding IDs and FEC Instance IDs are subject to IANA registration. For general guidelines on IANA considerations as they apply to this document, see [RFC5052]. IANA is requested to assign a value under the ietf:rmt:fec:encoding name-space to "Supercharged Code" as the FEC Encoding ID value associated with this specification, preferably the value 7.

9. References

9.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] "Secure Hash Standard", National Institute of Standards and Technology FIPS PUB 180-3, October 2008.

9.2. Informative References

[3] Timothy Vismor, "Matrix Algorithms."

[4] Sergio Pissanetzky, "Sparse Matrix Technology," Academic Press, London (1984).

[5] Timothy A. Davis, "Direct Methods for Sparse Linear Systems" SIAM, Philadelphia, Pa (2006)

[6] Yousef Saad, "Iterative Methods for Sparse Linear Systems" 2nd Ed. SIAM, Philadelphia, Pa (2003)

[7] I.S. Duff, A.M. Erisman, and J. K. Reid, "Direct Methods for Sparse Matrices" (2008) (ISBN: 978-0198534082)

[8] John K. Reid, "Solution of linear systems of equations: Direct methods" (1977)

[9] Golub, G.H. "Numerical methods for solving linear least-squares problems" Numerische Mathematik Volumne 7, Number 3 (1965) pp 206-216

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Erik Stauffer
Broadcom
190 Mathilda Place
Sunnyvale, Ca 94086

Email: eriks@broadcom.com

BZ Shen
Broadcom
5300 California Avenue
Irvine, CA 92617

Email: bzshen@broadcom.com

Soumen Chakraborty
Broadcom
RMZ Ecospace
Bellandur
Bangalore 560037, India

Email: soumen@broadcom.com

Djordje Tujkovic
Broadcom
190 Mathilda Place
Sunnyvale, Ca 94086

Email: djordje@broadcom.com

Jing Huang
Broadcom
190 Mathilda Place
Sunnyvale, Ca 94086

Email: jingh@broadcom.com

Kamlesh Rath
Broadcom
190 Mathilda Place
Sunnyvale, Ca 94086

Email: krath@broadcom.com