

Test and Evaluation/Science and Technology Program

AMT-40

NST “Middleware Enhancements for a Netcentric Simulation Architecture (MENSA)”

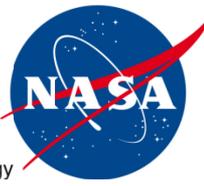
***Philip Tsao, Michael Cheng, Clayton Okino
{ptsao,mkcheng,cokino}@jpl.nasa.gov
Jet Propulsion Laboratory
California Institute of Technology***

December 18, 2008



Outline

JPL
Jet Propulsion Laboratory
California Institute of Technology



- **Introduction**
- **Source Coding**
 - Huffman Coding
 - Lempel-Ziv
- **Channel Coding**
 - LT Codes
- **Source and Channel Coding**
- **Results**
- **Acknowledgements**

Project Description

NST Gap Addressed

- Minimize congestive failure in NST environments.
- Overcome unreliable network environments in NST infrastructure.
- Dynamically optimize information delivery with graceful degradation in NST infrastructure.

Description

- Improving network efficiency (i.e. “blending” of the data)
- Improving reliability (using redundancy)
- Improving utilization (through dynamically coding)

CURRENT STATE

Minimal responsiveness to data flows for the synthetic battlespace that congest the link

Minimal responsiveness to data flows on open range environment that use unreliable links

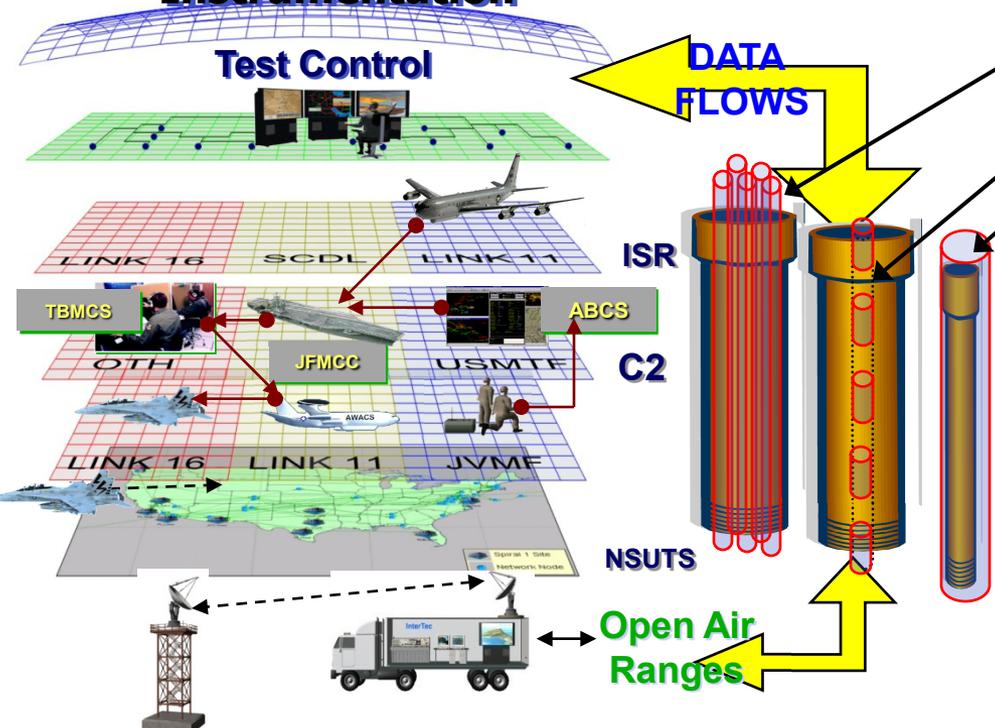
Minimal responsiveness to data flows to open range environment that use varying bandwidth links

Instrumentation

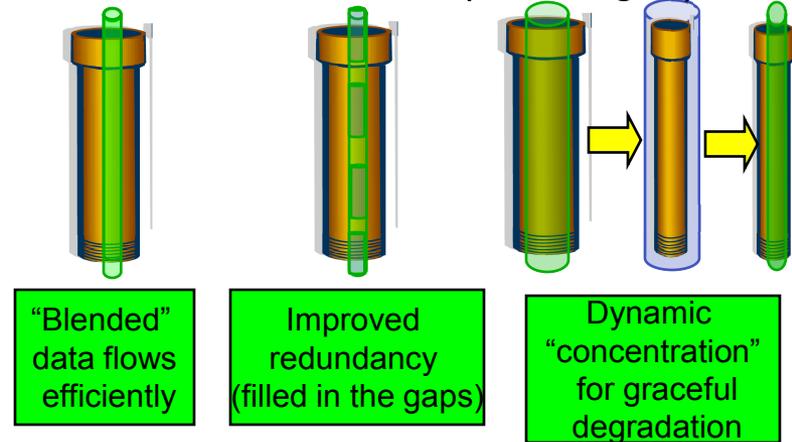
Test Control

DATA FLOWS

Joint End-to-End Mission Threads



PROJECT END STATE (lab testing of:)



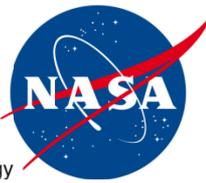
“Blended” data flows efficiently

Improved redundancy (filled in the gaps)

Dynamic “concentration” for graceful degradation



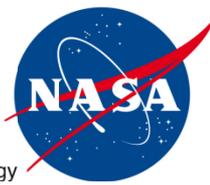
Source Coding algorithms



- **Lossy compression**
 - Quantization
- **Lossless compression**
 - Non-Uniform Distribution of Symbols -> Uniform Distribution of Fewer Symbols
 - Non-Uniform Distribution of Sequences of Symbols -> Uniform Distribution of Sequences of Fewer Symbols



Huffman Coding



- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

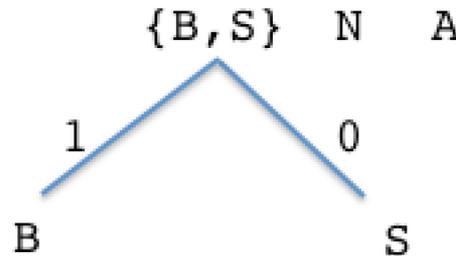
Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	

B S N A

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

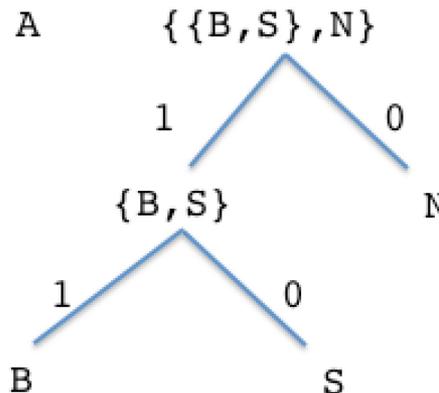
Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd Itr		
{B,S}	2/7	1
N	2/7	0
A	3/7	



Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0

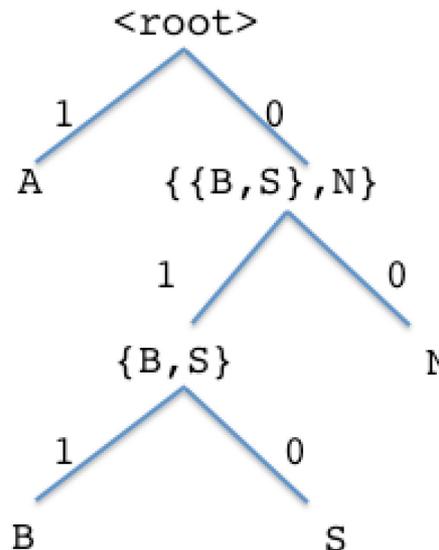


[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0

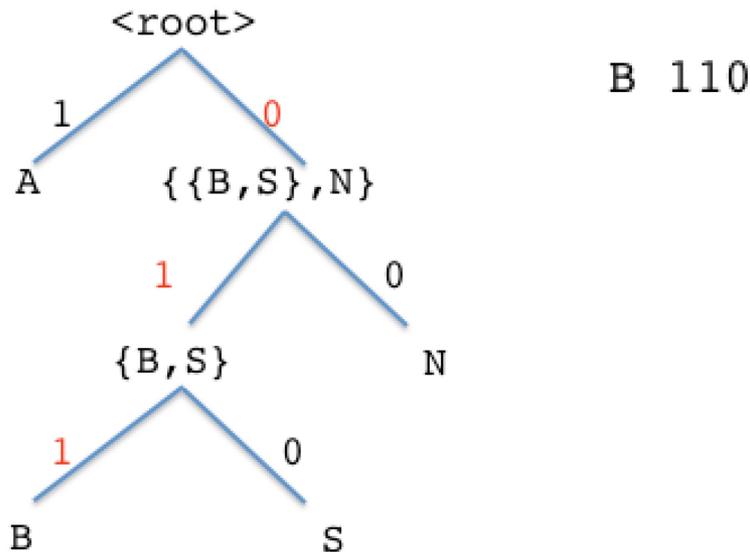


[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0

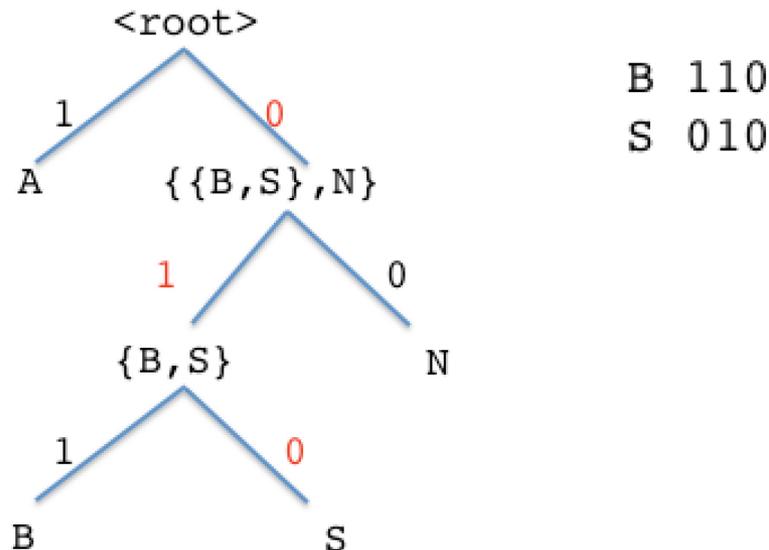


[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0

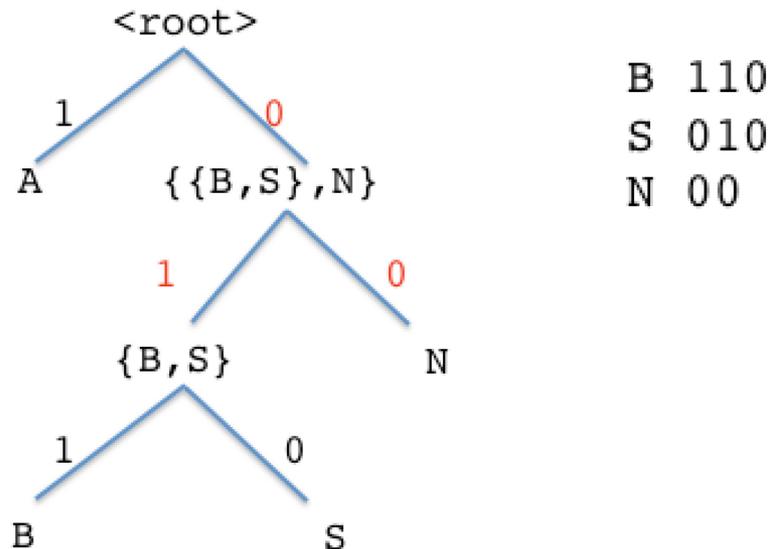


[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0

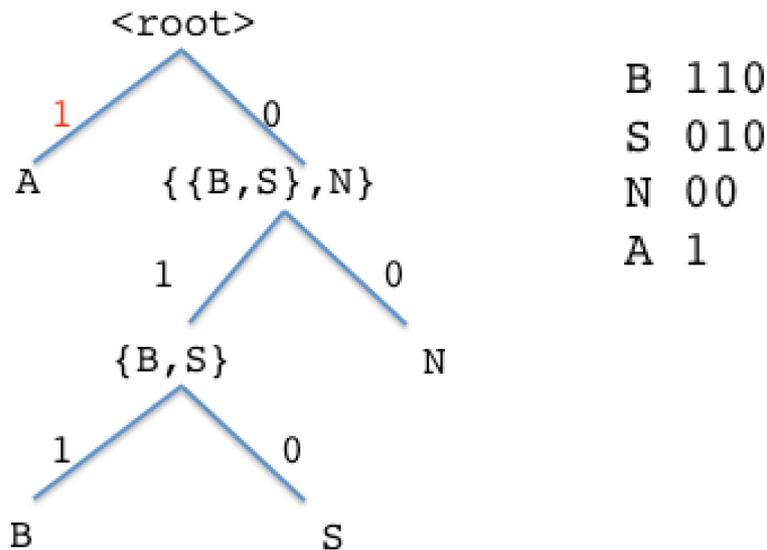


[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0

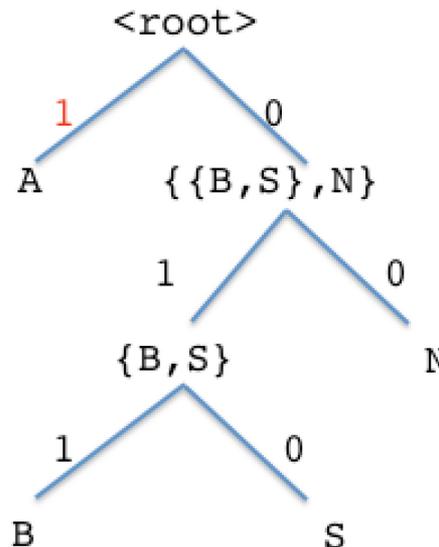


[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

Huffman Coding

- **Huffman encoding [HUFF] a given message by:**
 1. Sort the symbols of the message from lowest to highest frequency.
 2. Assign the lowest probability symbol the code "1" and the next lowest probability symbol the code "0".
 3. Merge the two lowest probability symbols into one symbol with the combined probability of the two symbols.
 4. Go to step 1 unless there are two symbols left.
 - The codeword for each symbol is determined by concatenating the codes encountered through traversing the resulting tree from the leaf node (symbol) to the root.

Symbol	Probability	Code
B	1/7	1
S	1/7	0
N	2/7	
A	3/7	
2 nd ltr		
{B,S}	2/7	1
N	2/7	0
A	3/7	
3 rd ltr		
A	3/7	1
{{B,S},N}	4/7	0



B 110
 S 010
 N 00
 A 1

 BANANAS
 1101001001010

[HUFF] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102

- **Lempel-Ziv [LZ] is a popular example of a dictionary coding algorithm in which references to a history buffer are transmitted in lieu of the uncompressed data.**
 - Symbols that are not found in the dictionary are represented explicitly. For example, the string BANANAS might be transmitted as “BAN(2,3)S”. The pair (2,3) denotes the string that begins with the 2nd to last symbol in the dictionary (in this case, “A”) that is 3 symbols long (ANA). Note that the third symbol of the string is not (yet) in the dictionary until the first symbol of the string is appended to the end of the dictionary.

History	Encoded	Decoded
	B	B
B		

[LZ] Jacob Ziv and Abraham Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, 23(3), pp.337-343, May 1977.

- **Lempel-Ziv [LZ] is a popular example of a dictionary coding algorithm in which references to a history buffer are transmitted in lieu of the uncompressed data.**
 - Symbols that are not found in the dictionary are represented explicitly. For example, the string BANANAS might be transmitted as “BAN(2,3)S”. The pair (2,3) denotes the string that begins with the 2nd to last symbol in the dictionary (in this case, “A”) that is 3 symbols long (ANA). Note that the third symbol of the string is not (yet) in the dictionary until the first symbol of the string is appended to the end of the dictionary.

History	Encoded	Decoded
	B	B
B	BA	BA
BA		

[LZ] Jacob Ziv and Abraham Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, 23(3), pp.337-343, May 1977.

- **Lempel-Ziv [LZ] is a popular example of a dictionary coding algorithm in which references to a history buffer are transmitted in lieu of the uncompressed data.**
 - Symbols that are not found in the dictionary are represented explicitly. For example, the string BANANAS might be transmitted as “BAN(2,3)S”. The pair (2,3) denotes the string that begins with the 2nd to last symbol in the dictionary (in this case, “A”) that is 3 symbols long (ANA). Note that the third symbol of the string is not (yet) in the dictionary until the first symbol of the string is appended to the end of the dictionary.

History	Encoded	Decoded
	B	B
B	BA	BA
BA	BAN	BAN
BAN		

[LZ] Jacob Ziv and Abraham Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, 23(3), pp.337-343, May 1977.

- **Lempel-Ziv [LZ] is a popular example of a dictionary coding algorithm in which references to a history buffer are transmitted in lieu of the uncompressed data.**
 - Symbols that are not found in the dictionary are represented explicitly. For example, the string BANANAS might be transmitted as “BAN(2,3)S”. The pair (2,3) denotes the string that begins with the 2nd to last symbol in the dictionary (in this case, “A”) that is 3 symbols long (ANA). Note that the third symbol of the string is not (yet) in the dictionary until the first symbol of the string is appended to the end of the dictionary.

History	Encoded	Decoded
	B	B
B	BA	BA
BA	BAN	BAN
BAN	BAN (2 , 3)	BANANA
BANANA		

[LZ] Jacob Ziv and Abraham Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, 23(3), pp.337-343, May 1977.

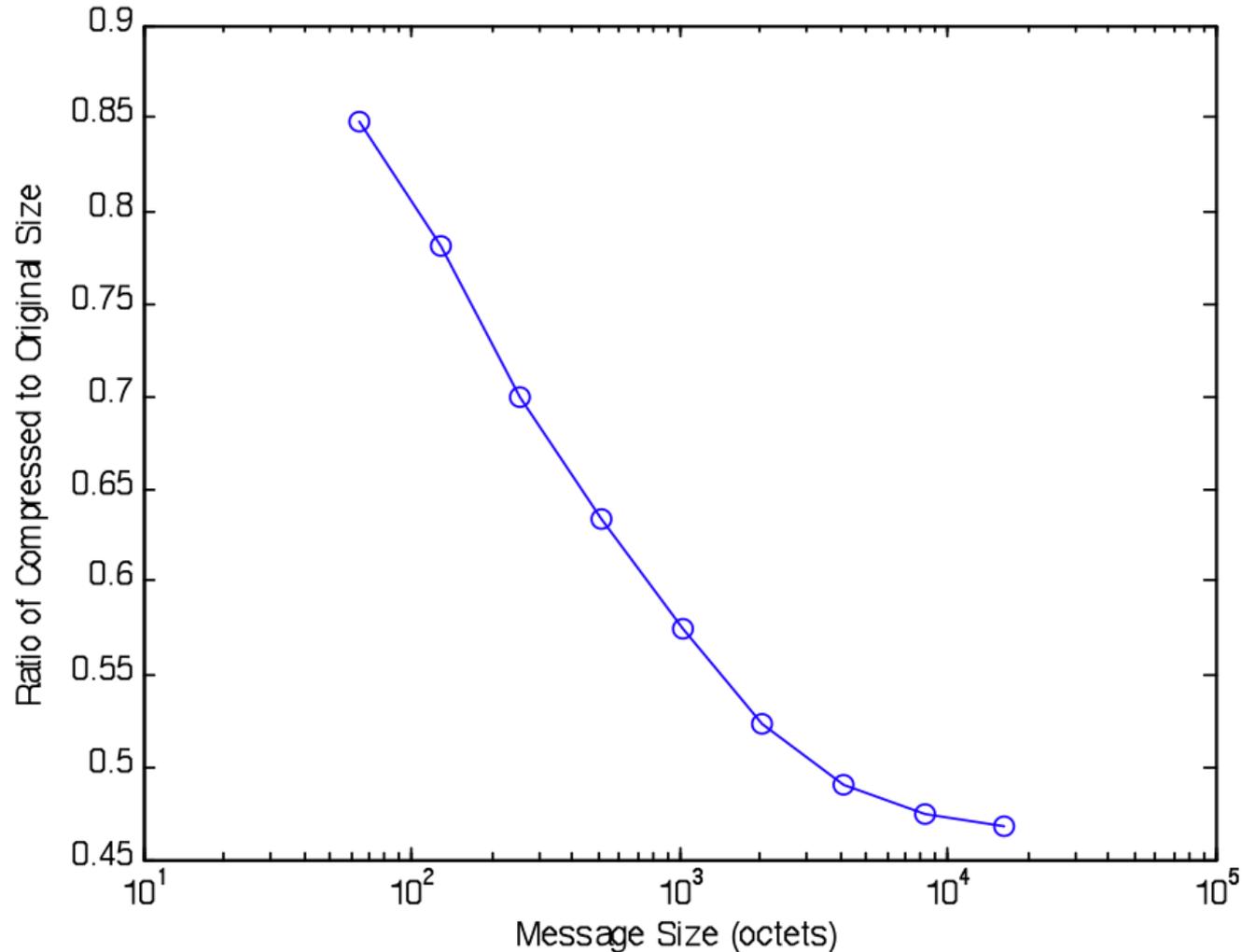
- **Lempel-Ziv [LZ] is a popular example of a dictionary coding algorithm in which references to a history buffer are transmitted in lieu of the uncompressed data.**
 - Symbols that are not found in the dictionary are represented explicitly. For example, the string BANANAS might be transmitted as “BAN(2,3)S”. The pair (2,3) denotes the string that begins with the 2nd to last symbol in the dictionary (in this case, “A”) that is 3 symbols long (ANA). Note that the third symbol of the string is not (yet) in the dictionary until the first symbol of the string is appended to the end of the dictionary.

History	Encoded	Decoded
	B	B
B	BA	BA
BA	BAN	BAN
BAN	BAN (2 , 3)	BANANA
BANANA	BAN (2 , 3) S	BANANAS

[LZ] Jacob Ziv and Abraham Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, 23(3), pp.337-343, May 1977.

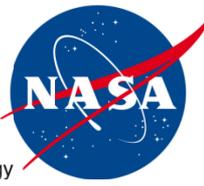


Compressibility of different length strings from the Calgary Corpus using a Lempel-Ziv Algorithm





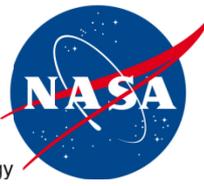
Channel Coding



- **Reducing redundancy (though source coding) reduces reliability**
- **Channel Coding adds redundancy**
 - Automatic Repeat Request (ARQ)
 - Forward Error Correction (FEC)



LT Codes

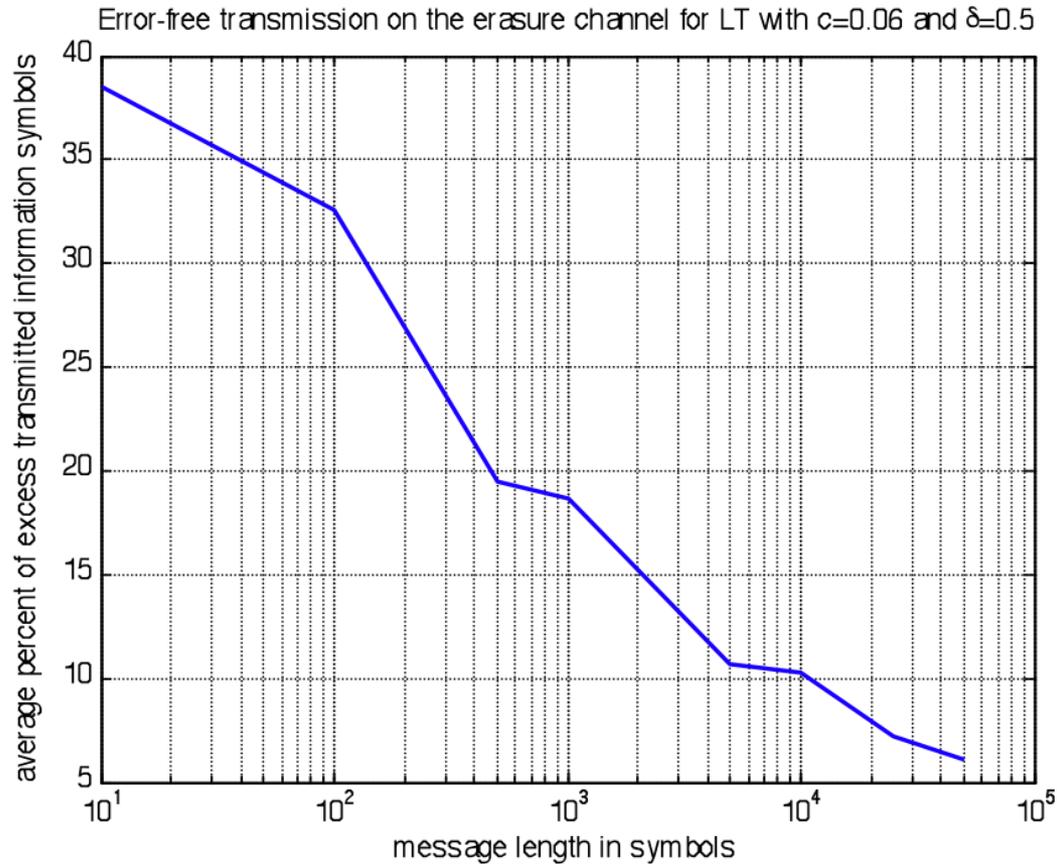


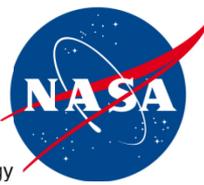
- **Randomly pick a set of packets from a fixed block of message packets according to a probability distribution and then linearly combine these selected packets into code symbols**
- **Rinse and repeat as needed**
- **If probability distribution is *Robust Soliton*, average excess transmission approaches zero as message blocksize increases**

[LT] M. Luby, "LT Codes," in Proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, pp. 271–282.

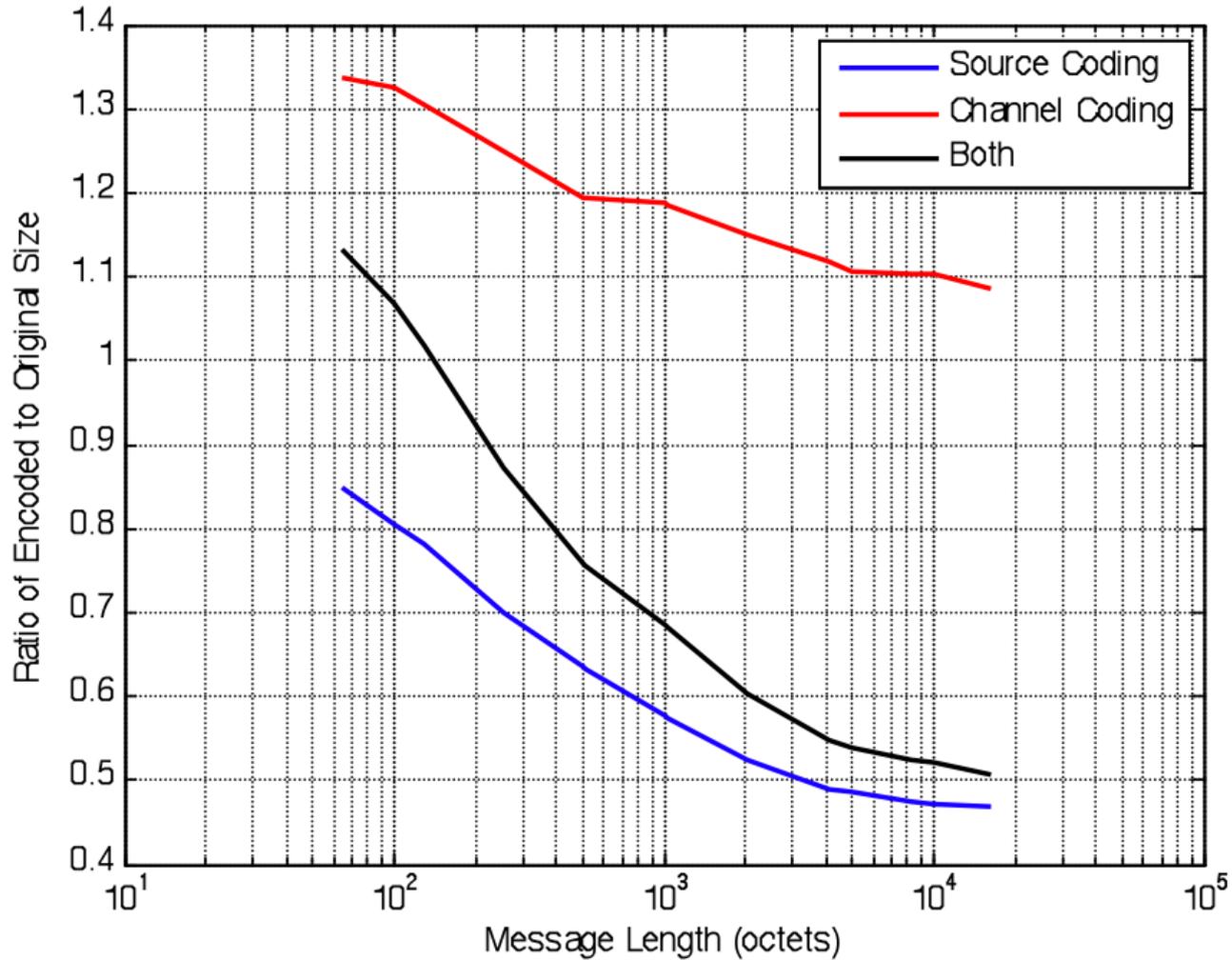
LT Codes

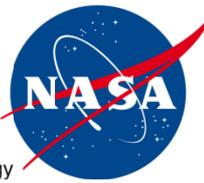
The average excess transmission required for error free operation diminishes with increasing message length.





- **Source Coding reduces redundancy**
 - Good
- **Source Coding reduces reliability**
 - Bad
- **Channel Coding increases reliability**
 - Good
- **Channel Coding increases redundancy**
 - Bad
- **Why not do *both*?**





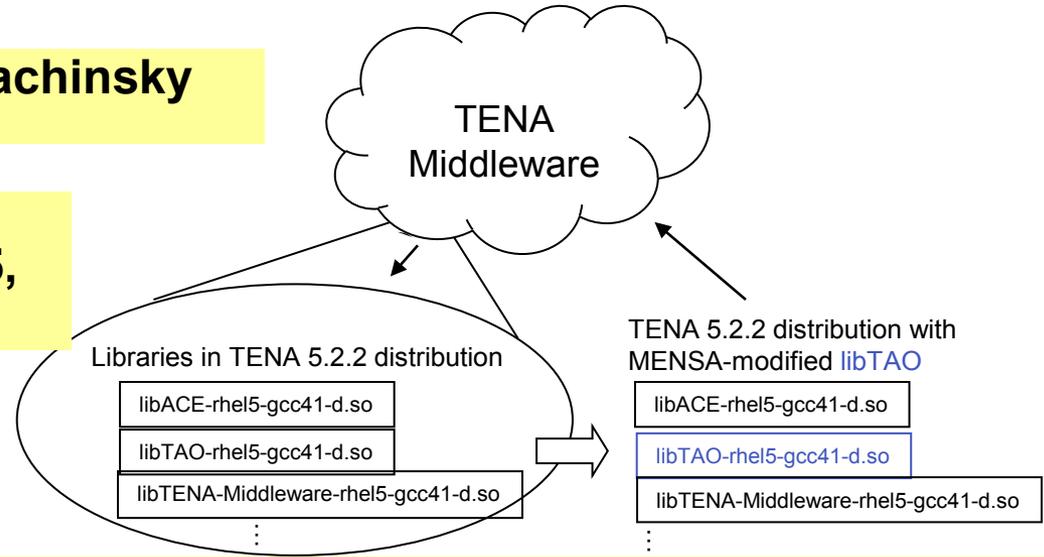
- **Not all forms of redundancy are equal**
- **Longer message lengths are better**
 - Source Coding efficiency increases
 - Channel Coding excess transmission decreases
- **Combination of Source and Channel Coding offers additional design tradeoffs**
 - Processing overhead vs. compression
 - Processing overhead vs. excess transmission
 - Compression vs. excess transmission

MENSA beta code in an unofficial TENA library

- **MENSA code sent to Steve Bachinsky at the end of November 2008.**

- **Aug 2008 – IOP compression patch available for Linux rhel5, fc5 and fc6.**

- **Various build Unix versions (e.g. rhel5-gcc41, rhel5-gcc41-d, fc5-gcc41, fc5-gcc41-d, fc6-gcc41)**
- **Different versions of TENA (e.g. 5.2.2 versus 6.0)**
- **Highlighted in BLUE are the versions planned to deliver under MENSA. Other variations will require additional funded effort.**

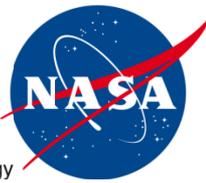


Delivered MENSA Beta versions to Pt. Mugu on August 29, 2008 for inclusion in NSTATIE laboratory. Eight versions were delivered. These are:

- xp-vc80 windows xp 32 bit
- xp-vc80-d windows xp 32 bit debug
- rhel5-gcc41 red hat enterprise linux 5
- rhel5-gcc41-d red hat enterprise linux 5 debug
- fc5-gcc41 fedora core 5
- fc5-gcc41-d fedora core 5 debug
- fc6-gcc41 fedora core 6
- fc6-gcc41-d fedora core 6 debug



Acknowledgements



Thanks to

JIM BAK

ELIZABETH BODINE

LOREN CLARE

SCOTT DARDEN

AARON KIELY

MATTHEW KLIMESH

SAMUEL NGUYEN

JEFFREY UNG

LIANG YU



Test and Evaluation/Science and Technology Program

Back Up

- April 2008 – Wireshark patch written to provide mechanism to evaluate MENSA algorithms on InterTEC platforms

The screenshot shows the Wireshark interface with a list of 18 captured packets. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Help), a toolbar with various icons, and a filter field. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	66.51.201.187	155.199.36.26	TCP	bcs > https [ACK] Seq=1 Ack=1 Min=62264 Len=0
2	0.062870	155.199.36.26	66.51.201.187	TCP	[TCP ZeroWindow] [TCP ACKed lost segment] http
3	4.999923	3com.66:3c:20	RedbackM_00:8d:92	ARP	Who has 66.51.201.1? Tell 66.51.201.187
4	5.038897	RedbackM_00:8d:92	3com.66:3c:20	ARP	66.51.201.1 is at 00:10:67:00:8d:92
5	13.988059	66.51.201.187	88.208.203.85	TCP	advant-lm > http [SYN] Seq=0 Min=5840 Len=0 Win=0
6	14.161459	88.208.203.85	66.51.201.187	TCP	http > advant-lm [SYN, ACK] Seq=0 Ack=1 Min=57
7	14.161722	66.51.201.187	88.208.203.85	TCP	advant-lm > http [ACK] Seq=1 Ack=1 Min=5840
8	14.162133	66.51.201.187	88.208.203.85	HTTP	GET /tor/server/d/2CRDA33R9215477D611C1CD3219
9	14.341338	88.208.203.85	66.51.201.187	TCP	http > advant-lm [ACK] Seq=1 Ack=218 Min=6912
10	15.828823	88.208.203.85	66.51.201.187	HTTP	HTTP/1.0 503 Directory busy, try again later
11	15.829029	66.51.201.187	88.208.203.85	TCP	advant-lm > http [ACK] Seq=218 Ack=49 Min=5840
12	17.047484	88.208.203.85	66.51.201.187	HTTP	Continuation or non-HTTP traffic
13	17.047655	66.51.201.187	88.208.203.85	TCP	advant-lm > http [ACK] Seq=218 Ack=1497 Min=86
14	17.055413	88.208.203.85	66.51.201.187	HTTP	Continuation or non-HTTP traffic
15	17.055553	66.51.201.187	88.208.203.85	TCP	advant-lm > http [ACK] Seq=218 Ack=2945 Min=11
16	17.230649	88.208.203.85	66.51.201.187	HTTP	Continuation or non-HTTP traffic
17	17.230801	66.51.201.187	88.208.203.85	TCP	advant-lm > http [ACK] Seq=218 Ack=4393 Min=14
18	17.237840	88.208.203.85	66.51.201.187	HTTP	Continuation or non-HTTP traffic

The detailed view of Frame 5 (74 bytes on wire, 74 bytes captured) shows the following layers:

- Ethernet II, Src: 3com.66:3c:20 (00:60:08:66:3c:20), Dst: RedbackM_00:8d:92 (00:10:67:00:8d:92)
- Internet Protocol, Src: 66.51.201.187 (66.51.201.187), Dst: 88.208.203.85 (88.208.203.85)
- Transmission Control Protocol, Src Port: advant-lm (2295), Dst Port: http (80), Seq: 0, Len: 0

The raw packet bytes are displayed in hexadecimal and ASCII format at the bottom of the interface.

- June 2008 – Wireshark tool predicts MENSA IIOP compression patch can achieve gains of 2-3X on InterTEC data (Performed by InterTEC network engineer)