Technology Brief

Analyzing the Molecular Sequence Reduction (MSR) Technology

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
408 745 2000 or 888 JUNIPER
www.juniper.net

Introduction

Repeated non-informative data patterns typically constitute between 60 and 90 percent of all network traffic. These repetitious patterns waste precious network resources and severely degrade network efficiency and application performance.

The patented Molecular Sequence Reduction™ (MSR™) compression technology — a major component of the Juniper Networks WX Framework™ which defines the attributes required of a comprehensive WAN optimization solution — enables networks to recover this previously wasted transmission capacity. The MSR technology dramatically improves WAN performance by detecting and transparently encoding repeated data patterns in a more efficient and compact form, increasing effective network capacity by three to 10 times.

This paper describes the key components of the MSR algorithm and analyzes the technology's most significant strengths and differentiators. The paper also compares the MSR technology to more traditional general-purpose data compression algorithms, most of which have been derived from the Lempell-Ziv dictionary-based compression algorithms [LZ77, LZ78].

The MSR Algorithm

The MSR algorithm operates by finding variable-sized repeating patterns anywhere in the data stream — even across multiple packets, applications and sessions. The MSR technology adaptively learns the patterns that occur most frequently in the data stream and assigns a unique internal label to each. By continuously learning new patterns, the MSR technology constructs a table of unique patterns and corresponding labels, analogous to a dictionary of the most commonly user words, phrases, sentences, etc. The MSR technology then uses this dictionary of patterns to encode the data stream in a highly efficient manner by replacing each subsequent instance of a specific detected pattern with its corresponding label. As a result, the numerous non-informative redundancies are stripped from the data stream before it is transmitted across a network's capacity-constrained wide-area network (WAN) links. A second MSR-enabled engine at the other end of the link receives the highly optimized data stream and converts it back to its original form by replacing each label with the original data pattern. As a result, no extra WAN resources are wasted in transmitting redundant data.

The MSR technology operates on network traffic in real time at the IP layer (Layer 3), where packets can be selected for MSR processing based on various parameters such as destination address. For instance, if a packet is destined to traverse an expensive and bandwidth-constrained link, and a second MSR-enabled device resides at the other end of this link, then it is selected for MSR compression.

Once a packet is picked up, all bytes within the packet — header as well as payload — are processed. Each byte, regardless of its source, is treated as an independent input symbol by the MSR technology. Since MSR-encoded packets can only be decoded by another MSR-enabled device, all processed packets are encapsulated in an IP tunnel for transmission; this "repacketizing" enables the MSR technology to reduce both the number of transmitted bytes as well as the number of packets that traverse the network.

Since all IP packets from any application, user or session can be processed and reduced, the MSR technology provides significant data reduction benefits to the entire network link. In addition, by performing its data reduction at the IP layer, the MSR technology remains completely transparent to the rest of the network, as well as applications, servers and clients. Table 1 shows some of the measured and confirmed reduction levels achieved by MSR technology on various live applications.

Application	MSR Data Reduction
E-mail	75 percent
Exchange	79 percent
MS SQL	80 percent
SAP	73 percent
Telnet	67 percent
Oracle	78 percent
File sharing	81 percent
Lotus Notes	78 percent
HTTP servers	96 percent
Search servers	84 percent
Image servers	70 percent

Table 1: Actual MSR results on a variety of live network traffic.

Dissecting the MSR Algorithm

The four primary components of the MSR algorithm are:

1. Pattern discovery
2. Adaptive dictionary construction
3. Dictionary synchronization
4. Data reduction

The following sections describe each of these components in more detail.

Pattern Discovery

The MSR technology's pattern discovery algorithm is one of its key differentiators. The speed and precision with which the pattern discovery algorithm searches for new patterns is critical to the high data reduction rates provided by MSR technology and the high network speeds at which this technology can operate.

As each new input symbol is processed by the MSR engine, it is first checked to determine whether it is part of a pre-existing pattern which has already been added to the dictionary. If the symbol is not part of a pre-existing dictionary pattern, the MSR pattern discovery algorithm searches a buffer consisting of previously processed symbols to determine whether the current symbol is part of a new pattern that occurred earlier in this buffer but has not yet been added to the dictionary. In order to maximize the amount of input data that can be stored in this historical "look-back" buffer, the buffer itself is encoded in a highly compacted format. As a result, the physical size of this buffer can be from tens to hundreds of megabytes, which can represent hundreds to thousands of megabytes of un-encoded data. The size of this historical buffer is determined by the amount of physical memory available on the device; the speed of the algorithm is not affected by size of the buffer. The MSR technology's ability to search this large buffer at high speeds is a critical strength of the algorithm and the reason for the high data reduction rates it achieves. The patterns discovered in this buffer are checked to see if they pass a significance threshold; if so, they are added to the dictionary by the dictionary construction module.

Adaptive Dictionary Construction

When a new pattern is discovered in the search buffer, it is submitted to the dictionary construction module, which "learns" the pattern and adds it to the dictionary by assigning it a unique internal label that will be used to refer to this pattern.

Since the number of dictionary labels is finite, the dictionary construction module also contains a "forgetting" component which effectively dissociates a label from an old pattern, allowing the label to be reassigned to a new pattern. Patterns that are forgotten or phased out are those that are least frequently used and therefore less relevant to the current data. Thus, while frequently used patterns (e.g. the word "the") could remain in the dictionary indefinitely, a less frequently used pattern (e.g., the word "xylophone") would only remain in the dictionary a short time and would soon be replaced by a more frequently occurring pattern.

The dictionary construction module adapts to the data that is being processed by maintaining only the most relevant and useful patterns in the dictionary. The learning and forgetting processes are both highly efficient and do not compromise the linearity or speed of the overall algorithm. The memory required to store the dictionary patterns is minimized by encoding the labels and patterns in a highly compact form so that, as in a true language dictionary, a relatively small table of patterns (tens of megabytes) can encode a large number of original data sequences (hundreds to thousands of megabytes). The fully adaptive nature of this process also ensures that the MSR technology can run continuously and incrementally on an indefinite sequence of input symbols.

Dictionary Synchronization

As described above, the MSR technology operates on both ends of a network link by encoding patterns with labels on one end, then decoding the labels and replacing them with the original patterns on the other end.

In order to guarantee that this process is completely loss-free and fully transparent, it is critical to ensure that all encoded labels are correctly decoded to their exact original patterns. This is achieved by the dictionary synchronization module through an independent communication channel established between each pair of MSR-enabled devices. As the encoding dictionary is incrementally updated, the synchronization module computes the significance of each dictionary pattern and incrementally communicates the relevant patterns to the corresponding decoding MSR device. Since the dictionary is incrementally updated, the communication overhead of this process is minimal and evenly distributed; synchronization does not require large periodic bulk transfers of the dictionary but rather schedules synchronization packets such that their overhead is functionally negligible.

Since the dictionary is maintained by an independent module and is synchronized through an independent communication channel, the data processing flow is decoupled from the dictionary synchronization process. The data reduction and reassembly processes therefore do not participate in the dictionary synchronization process, though they utilize the results of the dictionary synchronization to guarantee accurate encoding and decoding.

Since a single MSR-enabled device can communicate with multiple remote devices (i.e., a one-to-many as well as a many-to-one configuration), multiple encoding dictionaries can exist at the central "hub" device. The MSR technology utilizes a shared dictionary architecture on this central device in order to maximize the use of dictionary resources and provide greater end-point scalability. In this shared dictionary point-to-multi-point configuration, the synchronization module only transfers a pattern to a particular MSR-enabled device if the pattern is currently being used on this particular link. For example, if a headquarters site in San Francisco has links to remote offices in New York and Singapore, the MSR-enabled device in San Francisco will maintain a separate shared pattern dictionary for both remote offices, while the New York and Singapore MSR-enabled devices will only maintain pattern dictionaries that are relevant to their specific sites.

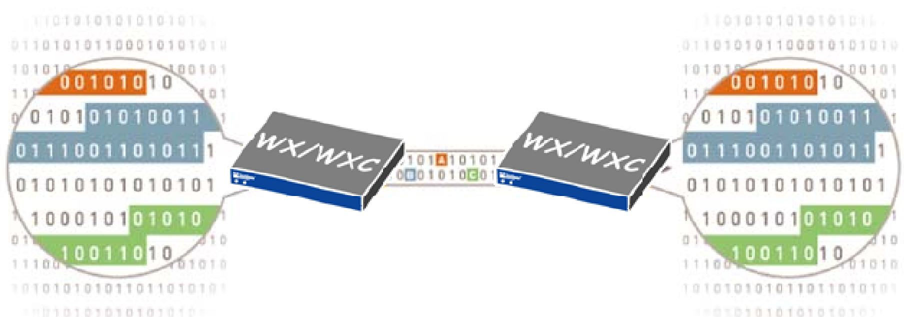

Figure 1: An illustration of the MSR technology's data reduction process.

Data Reduction

The data reduction component of the MSR technology operates directly on the data path by encoding individual packets based on its pattern dictionary. As each packet is processed, it is first examined symbol by symbol to determine whether it contains a pattern that already exists in the encoding dictionary. If no pre-existing dictionary pattern is found, the symbol is then checked by the pattern discovery module to see if it is part of a new pattern that should be added to the dictionary.

If a match to a pre-existing dictionary pattern is found (and the dictionary pattern has been correctly synchronized with the corresponding decoding MSR-enabled device), the data reduction process replaces the pattern in the data stream with the corresponding label for this pattern. This process is carried out on each individual packet, independently of all other packets.

Since the dictionary synchronization is performed through a separate process, the MSR technology does not impose any dependencies between processed packets, allowing the data reduction process to operate at high speeds and to minimize processing latency. After the last symbol in the packet is processed, the data reduction module immediately generates all of the encoded output for the packet and prepares the encoded packet for encapsulation and transmission.

Though the MSR technology operates across sessions by finding patterns across all packets in the data stream, the pattern replacement is performed only within individual packets. In addition to minimizing processing latency, this also allows each reduced packet to be treated independently of all other packets. Thus packets that are encoded by an MSR-enabled device can be transmitted to the decoding device in a connectionless fashion without imposing any ordering or reliability constraints on the encoded data stream.

Since packets that are encoded by an MSR-enabled device must be decoded by a specific decoding MSR-enabled device, the encoded packets are transmitted to the decoding device in an encapsulated form. Since the entire input packet is encapsulated, the header as well as payload of the packet can be encoded by MSR technology. In addition to reducing the number of transmitted bytes, the MSR technology can also reduce the number of transmitted packets by encapsulating multiple input packets into a single output packet. MSR technology minimizes the latency of this encapsulation and packet reduction process by setting a maximum limit of "*T*" milliseconds for encapsulation. The threshold *T* is set by default to 2 ms but can be automatically adjusted based on the round-trip time (RTT) across the network.

MSR Strengths and Differentiators

The MSR technology can be compared to traditional general-purpose data compression technologies since they both reduce data or object sizes by removing repetitions in the traffic stream. Therefore, the objective of both technologies is the same: find and eliminate data repetitions.

The efficacy of a compression technology is typically measured in terms of the data reduction rates it can achieve. But applying data compression techniques to the networking infrastructure imposes additional constraints that must be considered when evaluating the technology. These constraints include real-time operability, speed, latency, dependencies between encoded packets, transmission of encoded data through a non-reliable (lossy) medium, and incremental and continuous operation. This section examines MSR technology and traditional general-purpose data-compression techniques with respect to each of these issues, as well as their impact on data reduction and scalability. The data compression techniques used for comparison purposes are those derived from the Lempel-Ziv algorithms [LZ77, LZ78]. Many of the most widely known compression techniques, including gzip, compress, LZW and LZS (Stac compression) emerged from this original work by Lempel and Ziv.

The primary strengths of the MSR technology that differentiate it from traditional compression techniques are:

1. Linear data reduction and processing scalability
2. Decoupled data flow and dictionary synchronization
3. Incremental and continuous functionality

The following sections discuss each of these features in detail and compare them to those of traditional LZ-based compression techniques.

Linear Data Reduction and Processing Scalability

The degree of data reduction that can be achieved by a compression algorithm is a direct result of the number of repeated patterns it can discover. The number of detected pattern repetitions depends on the size of the buffer that stores copies of previously processed input symbols and the effectiveness of the search algorithm that looks for repetitions in this buffer.

The size of the historical input buffer directly influences the number of patterns that can be discovered and hence the degree of data reduction that the algorithm can achieve. In order for a repeated pattern to be detected, the number of bytes that occur between two subsequent repetitions of the pattern must be smaller than the buffer size — i.e., both instances of the repetition must fall within the buffer. For example, let us assume:

D = average distance between any two repetitions of a pattern
B = size of the search buffer

"D = 100 bytes" implies that a given pattern is repeated, on average, every 100 bytes in the input data. In order to efficiently detect a repetition of this pattern, B must clearly be larger than 100 bytes. If B is equal to 100 bytes plus two times the size of the repetition, the probability of detecting any given repetition of the pattern is 50 percent. If B is made smaller, the probability of detecting the repetition decreases; if B is made larger, the probability increases, approaching 100 percent. Therefore, the larger the buffer, the larger the number of data patterns that will be detected; the larger the number of data patterns that will be detected, the greater the data reduction.

Based on the above analysis, compression algorithms should clearly try to maximize the buffer size B in order to maximize their data reduction rate. But as B increases, so does the number of bytes or symbols that must be examined to search for pattern repetitions. Thus, both the data reduction rate and the processing speed of a compression algorithm must be examined simultaneously.

The key factor that determines the speed of a data reduction technique is its *complexity*, which is defined in terms of the number of CPU operations required to process an input sequence of "*n*" symbols. The complexity of the MSR algorithm is *O(n)* or "*Order(n)*." This indicates that the number of operations required to process *n* input bytes is proportional to *n*. The MSR technology is therefore a *linear* algorithm since the number of CPU operations required to process the input grows linearly with the number of input symbols. Thus, if the data transmission rate on a network link were doubled, the MSR technology would need to perform twice the number of operations in the same time in order to process all the data on the link. The MSR pattern discovery algorithm is the key component that provides this linear scalability. The pattern discovery algorithm is capable of searching any arbitrary-sized buffer in constant time, ensuring that every input symbol incurs a constant number of operations. As a result, the size of the search buffer, B, does not appear in the complexity equation of MSR technology. Furthermore, since the reduction rate of the MSR algorithm is proportional to the size of the buffer, we can conclude that the speed of the MSR algorithm is independent of the reduction rates that it can achieve. In other words, the MSR technology's buffer size, B, can be made arbitrarily large in order to achieve higher data reduction without impacting the speed or complexity of the algorithm.

In contrast to MSR technology, where the search buffer size and speed are independent variables, these variables are inversely related for most traditional LZ-based compression techniques. Hence, traditional compression techniques must sacrifice speed in order to achieve higher data reduction rates or sacrifice data reduction in order to achieve higher processing speeds. This is due to the fact that the complexity of these compression algorithms is $O(n \cdot f(B))$, where n is the number of input symbols and B is the size of the search buffer. This indicates that the number of operations required to process a sequence of n input symbols is proportional to the product of n and a function of the buffer size B. If the buffer size B is increased to achieve higher data reduction, the number of operations required to process n input symbols also increases. This in turn reduces the total speed and throughput of the compression algorithm.

Due to the inverse dependency of speed to buffer size, traditional LZ based compression techniques usually utilize a very small search buffer. Offline compression algorithms such as gzip typically utilize a buffer size of 8KB to 32KB. Online compression algorithms that must operate in real-time on live network traffic utilize even smaller buffers, typically 2KB or less. Additional networking constraints (described in the following section) further limit traditional compression techniques by forcing them to operate independently on each individual packet. In other words, each packet is compressed independently of all other packets and the compression algorithm resets its state after each packet is processed. Since the search buffer size cannot exceed the size of the input data that is being processed, traditional network compression techniques are often limited to a buffer size of 1500 bytes (i.e., the MTU of the network).

The scalability and data reduction of the MSR technology and traditional compression can be further explored by modeling the distance between repetitions in network traffic. Network links are typically upgraded to higher speeds to support higher data transfer rates, more users, and more applications. As a result, higher-speed links typically support a greater volume and variety of traffic. This further implies that at higher speeds, the average distance between repeated objects also increases (i.e., as network speeds increase, the average number of bytes that occur between two instances of the same pattern increases). We can therefore model the distance between repetitions as a normally distributed random variable, where the mean and standard deviation are a function of the link speed. For the current analysis, the mean distance between repetitions is modeled to be 20 percent of the number of bytes transferred per second and the standard deviation is 10 percent of byte transfer rate (i.e., half of the mean).

Mean: μ = (bytes per second) * 0.2
Standard deviation: σ = (bytes per second) * 0.1

Based on this model, we can accurately compute the probability of a search buffer of size B capturing two instances of a repetition. For example, if B is equal to the mean, μ, then on average half of the repetitions will fall within the buffer and half will fall outside of the buffer (assuming for modeling purposes that the size of the repetition is negligible compared to the distance between repetitions). Thus for $B = \mu$, the percentage of repetitions that will be detected is 50 percent. For all values of B, the pattern detection percentage is calculated as the area under the standard normal distribution curve less than Z, where Z is defined as:

$$Z = \frac{B - \mu}{\sigma}$$

Table 2 shows the buffer size B for both MSR technology and traditional compression at various network speeds. Since MSR technology can increase the buffer size without impacting algorithm complexity, the buffer size for each link speed is set as:

$$B_{MSR} = \mu + 2*\sigma$$

Hence, the Z score for MSR technology is always:

$$Z_{MSR} = \frac{B_{MSR} - \mu}{\sigma} = 2$$

A constant Z score of 2 implies that the probability of a repetition occurring within B_{MSR} bytes is 0.97, i.e., the percentage of repetitions that will be detected by MSR technology is a constant 97 percent.

Due to the limitations of traditional compression algorithms, the buffer size, B, for these techniques is kept constant at 1500 bytes. Thus for each link speed, the percentage of patterns that are detected is computed based on the following Z score:

$$Z_{Compression} = \frac{1500 - \mu}{\sigma}$$

The probability of any two repetitions occurring within 1500 bytes is then calculated in each case as the area under the standard normal curve less than $Z_{Compression.}$

Speed	Distance Between Repetitions (in bytes)		Buffer Size (in bytes)		% Repetitions Detected	
	Mean (μ)	Std. Dev. (σ)	MSR	Traditional Compress.	MSR	Traditional Compress.
56 Kbps	1434	717	2867	1500	97.7%	53.7%
128 Kbps	3277	1638	6554	1500	97.7%	13.9%
256 Kbps	6554	3277	13107	1500	97.7%	6.1%
512 Kbps	13107	6554	26214	1500	97.7%	3.8%
1.54 Mbps	40370	20185	80740	1500	97.7%	2.7%
45 Mbps	1179648	589824	2359296	1500	97.7%	2.3%
155 Mbps	4063232	2031616	8126464	1500	97.7%	2.3%
622 Mbps	16305357	8152678	32610714	1500	97.7%	2.3%

Table 2: The percentage of repetitions detected by MSR technology and traditional compression at various data rates.

Figure 2 shows the results from this analysis in a graphical format. Note that the results show only the percentage of repetitions that are detected and not the overall data reduction achieved. The overall data reduction can be computed as a product of the rate at which the repetitions occur in the data and the percentage of these repetitions that are detected. Thus if the rate at which repetitions occur is 80 percent and the percentage of detected repetitions is 97.7 percent, the overall data reduction will be approximately 78 percent.

For certain types of data (such as ASCII text), both the MSR technology and traditional compression techniques can provide additional data reduction by using statistical coding schemes to further compress the output packets. These statistical coding techniques are computationally expensive and only benefit a very limited set of data types. Thus, if the data reduction achieved by eliminating repetitions is high (as is the case with MSR technology), the incremental value of this statistical coding is marginal and does not justify the added computational cost. On the other hand, since the percentage of detected repetitions is low for traditional compression techniques, they must rely heavily on statistical coding to provide additional compression for certain limited data types.

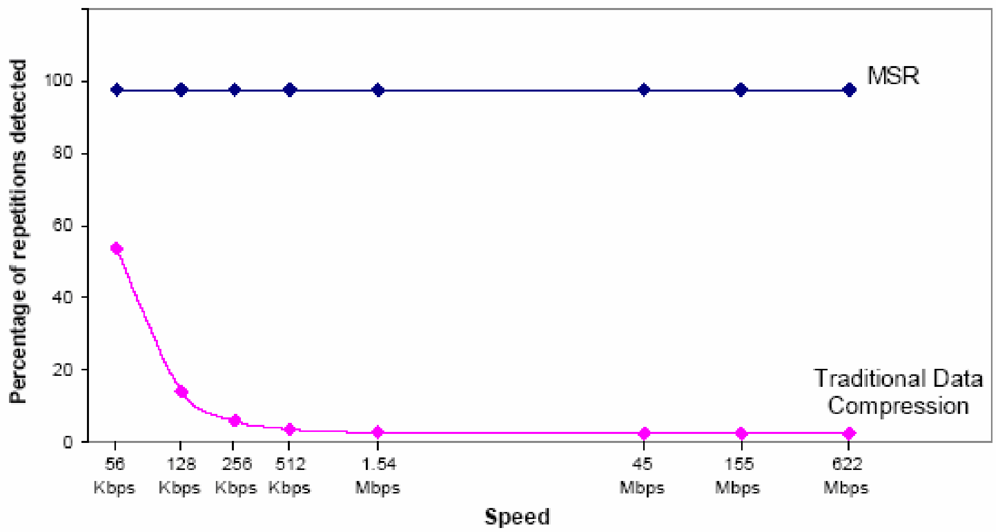

Figure 2: Comparing the scalability and data reduction of MSR technology and traditional data compression techniques.

Decoupled Data Flow and Dictionary Synchronization

A second major strength of MSR technology is the technique it uses to maintain and synchronize its pattern dictionaries. The MSR dictionary consists of an explicit table of patterns and associated labels; this dictionary is itself encoded in an optimized data structure that allows maximum pattern storage in a small amount of physical memory. This optimized encoding structure also minimizes dictionary synchronization overhead by allowing the encoding MSR device to transmit only small incremental dictionary updates to the decoding device.

Since the MSR dictionary is maintained and updated explicitly by an independent module, the data processing components are decoupled from the dictionary synchronization process. Though the data reduction and re-assembly processes require dictionary synchronization to guarantee accurate encoding and decoding, they do not directly participate in dictionary synchronization. The encoded and encapsulated packets that are transmitted from the encoding MSR device to the decoding device therefore do not have any dependencies between them. This implies that the reliable and in-order delivery of these encapsulated packets is not a requirement for dictionary synchronization and accurate data decoding.

This is a key strength of the MSR technology since it allows each output packet to be encapsulated in an independent connectionless packet. If an encapsulated packet is lost during transit, or if the encapsulated packets are delivered to the decoding device out of order, the decoding MSR engine will not be affected and will continue to accurately decode all packets. Since end-to-end communication is typically based on reliable transport protocols such as TCP, encapsulating compressed packets in a second TCP channel can significantly impact network performance. Nesting multiple

TCP control loops within each other can cause unstable network behavior and significantly degrade performance. Therefore, the MSR technology's ability to perform data reduction across all packets and yet transmit each processed packet via a non-reliable transport channel is a significant differentiator.

As described in the previous section, for performance reasons, traditional data compression techniques are often limited to small buffer sizes of about 2KB. Hence, these techniques treat each input packet independently of all other packets and do not look for repetitions across multiple packets. Even if traditional compression techniques were able to overcome this performance limitation and look for repetitions across multiple packets, they would then be faced with inter-packet dependencies.

Most LZ-based data compression techniques use a pointer-based repetition encoding scheme that encodes repeated patterns by pointing to the previous position in the input stream where the pattern occurred. For example, if a pattern in the packet that is currently being processed had occurred in the previous packet, the compression algorithm would encode the current instance of the pattern by pointing to its precise location in the previous packet. As a result, the compression algorithm would require that both packets arrive at the decoding device in the same order in which they were transmitted and that neither packet be lost during transport. This can only be achieved by encapsulating all compressed packets via a reliable transport protocol such as TCP.

As described above, encapsulating compressed packets in a TCP channel can cause unpredictable network behavior due to the nesting of multiple TCP control loops. At the same time, since many independent user sessions would be transported through a single encapsulation TCP session, TCP's congestion control mechanisms could significantly reduce bandwidth utilization and further degrade performance. Due to these factors, traditional LZ-based compression techniques cannot encode repetitions across packets and must reset their state after each packet. This further reinforces the single packet limitations of traditional compression techniques described in the previous section.

Incremental and Continuous Functionality

A third significant differentiator of the MSR technology is its ability to operate incrementally and in real time on a continuous stream of data. MSR technology processes each input symbol incrementally and generates output while continuing to process more input data. Non-incremental algorithms typically process large blocks of input at a time and can only generate output after a complete block has been processed. The incremental nature of the MSR technology is an important feature since it avoids the additional latency incurred by waiting for a fixed amount of input data to be processed before generating the reduced output. Though MSR technology searches for and eliminates repetitions across packets, the encoded output for an input packet is available for transmission as soon as the input packet has been processed. No additional latency is introduced for each input packet.

MSR technology can operate continuously within finite resources on an indefinite stream of input data. All resources, such as physical memory and dictionary space, are optimally utilized and re-used to ensure that the algorithm never exceeds its allocated limits. The re-use of dictionary resources is performed through an adaptive "forgetting" process that replaces infrequently utilized patterns with more relevant and recently discovered patterns.

The resource management modules do not impact the linearity or overall scalability of the MSR algorithm in any way. The MSR technology's adaptive dictionary learning and forgetting processes can operate continuously while still maintaining the linear complexity and scalability of the algorithm.

Conclusions

The Juniper Networks MSR algorithm is a linearly scalable technology that can provide consistently high repetition detection and data reduction rates as network speeds increase. This is due to the fact that the MSR technology's search buffer for discovering repeated patterns can be increased arbitrarily without impacting the speed of the algorithm. Thus, as network speeds increase and the distance between repetitions becomes larger, the MSR technology's search buffer size can be increased to capture these distant repetitions without affecting the computational complexity of the algorithm.

In contrast to the MSR technology, traditional data compression techniques are not scalable to high speeds since they are limited to a fixed buffer size. These techniques have an inverse relationship between the speed of the algorithm and buffer size, and therefore cannot increase the search buffer to suit higher data rates. In addition, these compression algorithms typically impose strong dependencies across the compressed data stream, which requires the algorithms to reset their state after each input packet. Due to these factors, traditional compression techniques operate on each input packet independently of all other packets and cannot detect repetitions across multiple packets.

An analysis of the scalability of the MSR technology and traditional compression techniques, determined by modeling the distance between repetitions as a normally distributed random variable, found that the average distance between repetitions was 20 percent of the number of bytes transmitted per second. Hence, as network speeds increased, the average distance between repetitions also increased. The results of this analysis show that MSR technology can consistently achieve a high repetition detection rates of 97 percent at all speeds by increasing the search buffer size to suit the higher data rates. In contrast, the repetition detection rate for traditional compression techniques drops from 54 percent at 56 Kbps to two percent at 622 Mbps. This dramatic drop in repetition detection is due to the fact that the search buffer for traditional compression is limited to a constant size and cannot be increased at higher speeds.

The MSR algorithm decouples the data flow from the dictionary synchronization process and does not impose dependencies across the encoded and encapsulated output packets. Though the algorithm provides significant data reduction across all input packets, each output packet can be independently transmitted to the decoding device through a standard non-reliable transport protocol. The MSR technology is also capable of running incrementally and continuously on an indefinite stream of input data. Dictionary resources are continuously and efficiently re-used to ensure that the algorithm always remains within fixed memory limits. The MSR technology adaptively learns and forgets patterns in order to maintain a relevant set of patterns in its dictionary, and does so without compromising the overall linear complexity of the algorithm.

References

[LZ77] Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337-343.

[LZ78] Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable rate coding. *IEEE Transaction on Information Theory*, IT-24(5):530-536.