SCRAMBLING AND 4LZS WITH RUN SUBSTITUTION

by

Richard Cam, Ph.D.

PMC-Sierra, Inc.
105-8555 Baxter Place
Burnaby, B.C.
Canada V5A 4V7

Telephone: (604) 415-6000
Direct: (604) 415-6022
FAX: (604) 415-6205
E-mail: richard_cam@pmc-sierra.bc.ca

21 Oct. 1996

Abstract

In this report, the performance of 4LZS with run substitution used in conjunction with scrambling is evaluated. Run length statistics corresponding to random bit streams are calculated for two run substitution schemes.

I. Introduction

Scrambling is used to spread the spectrum of transmitted data so as to reduce EMC problems associated with broadband data transmission over unshielded media. Quasi-periodic patterns in the source data stream can introduce emission spikes. By randomizing these patterns through scrambling, the emission spectrum is smoothed out. Scrambling can also improve data transmission characteristics by reducing jitter and inter-symbol interference. Runs (sequences of unchanging bits) are also broken up by scrambling, resulting in a more reliable stream for clock recovery. Scrambling is also used to provide some measure of link security.

In order for the receiver to descramble the received data, it must be synchronized with respect to the transmitter's scrambling sequence. Accordingly, scrambling schemes can be classified by the method of synchronization. There are three categories: frame synchronous, self-synchronous and distributed sample scrambling. In frame synchronous scrambling, the scrambler is initialized at the start of frame transmission. The descrambler at the receiving end is likewise initialized at the

same reference point. Frame synchronous scrambling is susceptible to malicious "killer packet" attacks, wherein packets are constructed in such a way as to produce a constant output stream after scrambling, thus causing the receiver to (eventually) lose symbol-level sync.

With self-synchronous scrambling, the scrambling pattern is generated from the transmitted bit stream itself. As its name implies, the received bit stream automatically acquires (descrambling) synchronization when the number of bits received equals the memory interval of the scrambling scheme. Self-synchronous scrambling can be used to foil killer packet attacks, but error multiplication is introduced, whereby even a single bit error in the received scrambled bit stream can result in multiple bit errors at the descrambler's output. This problem can arise because the value of any given scrambled bit is dependent upon the values of other bits in the transmitted stream.

Distributed sample scrambling can be viewed as a variation of frame synchronous scrambling, where the scrambler can operate continuously without being reinitialized at the start of each frame. Samples of the scrambling bits are sent to the receiver for synchronization. If the samples match corresponding descrambling bits, the receiver assumes synchronization has been attained and leaves its descrambler's state unchanged. If the samples do not match on the other hand, then a correction algorithm is applied to the descrambler's state to (eventually) bring it into sync with the received scrambled stream. While frame synchronous scrambling is sufficient for randomizing long frames (e.g., SONET/SDH, Ethernet), distributed sample scrambling produces better scrambling characteristics for short frames (e.g., ATM cells). Self-synchronous scrambling is suitable for short and long frames, but at the expense of error multiplication. Distributed sample scrambling is not susceptible to error multiplication (in the scrambled stream) but is sensitive to errors in the transmitted samples. The need to send samples may also result in bandwidth expansion, although it can be avoided depending on the application at hand.

II. Examples of Scrambling

Frame synchronous scrambling is specified for 100Base-Tx Ethernet in the ANSI X3.263-1995 TP-PMD standard [1]. The generating polynomial is given by $1+x^9+x^{11}$. As shown in Fig. 1, an 11-stage linear feedback shift register (LFSR) is used to generate the cipher stream, whose output is XOR'ed onto the transmit data stream. The TP-PMD scrambler has a period of 2047 bits (i.e., indicating an efficient design, in which all possible states of the 11-bit register are visited, except of course for the all zeroes pattern, which produces an all-zero sequence). The scrambler's register is initialized to 00...001 at the start of frame transmission. Frame synchronous scrambling is also used for scrambling SONET frames. The generating polynomial is $1+x^6+x^7$, and has a period of 127. The scrambler is initialized to 1111111 at the most significant bit of the byte following the STS-1 number N C1 byte. The scrambler's

output is tapped from the x^7 position and added modulo 2 onto this bit and all subsequent bits throughout the complete STS-N frame [2].

Distributed sample scrambling is specified for the cell-based PHY layer in ATM [3]. The generating polynomial is $1+x^{28}+x^{31}$. The entire cell is scrambled except for the HEC field. In order to recalculate the HEC CRC for the first 32 (scrambled) bits in the header, the same HEC CRC is calculated on the 32 scrambler bits coincident with the first 32 header bits. This CRC is then added modulo-2 to the original HEC CRC. Two samples from the scrambling sequence are added (modulo 2) to the seventh and eighth bits of the HEC byte. Consequently, only the last six HEC bits can be used for cell delineation, since the modified bits cannot be used for error control until the descrambler is synchronized.

For SDH-framed ATM on the other hand, a self-synchronous scrambler with generating polynomial $1+x^{43}$ is recommended [3, 4]. Only the payload area of the ATM cells are scrambled to avoid error multiplication in the cell header (the headers are sent unscrambled; (de)scrambling operations are suspended and the (de)scrambler states are preserved during these 5-byte intervals). Note that if a bit error occurs in the scrambled transmission stream, then another bit error will be introduced 43 bit times later when a (hitherto correct) bit is XOR'ed with the output of the scrambler (propagating the original errored bit). The ATM cells are mapped onto the payload area of a SONET/SDH frame, which itself is (frame-synchronous) scrambled.

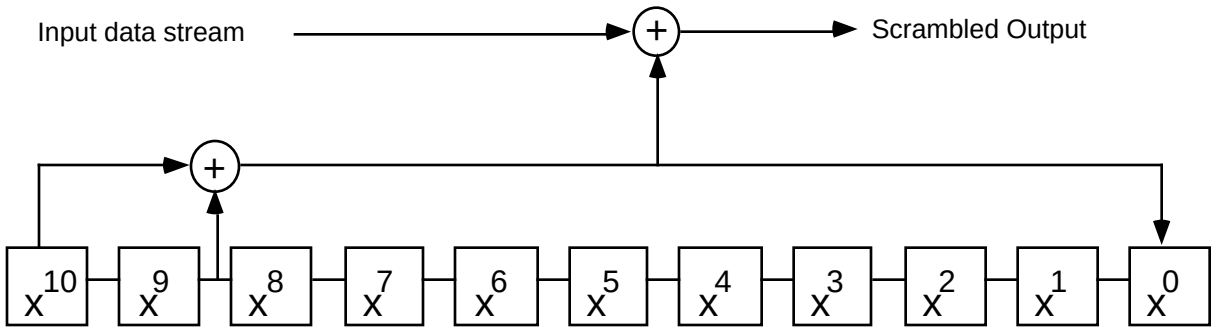

Fig. 1. Block diagram of TP-PMD scrambler.

III. Run Substitution and Run Lengths.

Since 2 bits are encoded into 1 4LZS symbol, there are four possible repeating patterns, taken two bits at a time, that will generate constant symbol runs, as shown in Table 1. Two run substitution schemes are considered in this report. In the first scheme, hereafter referred to as slotted run substitution, substitutions can occur only at fixed periodic boundaries that delineate slots of length equal to the run

substitution threshold. In this scheme, only runs which completely fill a slot are substituted. Hence, substitution codes are inserted at well-defined boundaries, which can increase robustness against channel errors. In the second scheme, substitutions are made whenever the run threshold is reached, regardless of slot alignment. This scheme, referred to as unslotted run substitution, bounds the run length more tightly than the slotted scheme but may be less robust since the substitutions can occur at arbitrary positions. Examples for these two schemes are depicted in Fig. 2.

Table 1. Constant run patterns with 4LZS coding.

Bit Stream	Quat Stream
101010.....	+3 +3 +3 .....
111111.....	+1 +1 +1 .....
010101.....	-1 -1 -1 .....
000000	0 0 0 .....

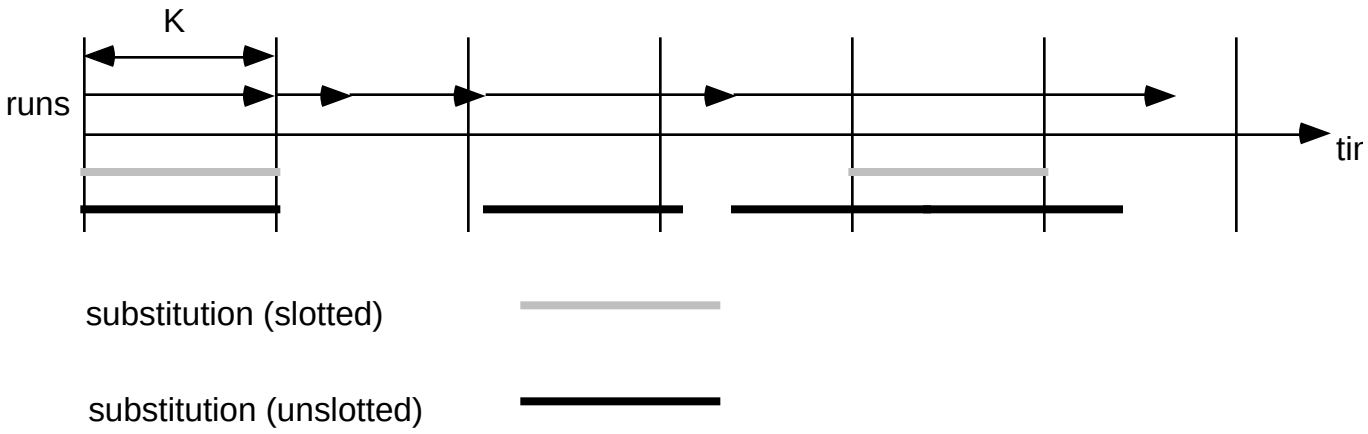

Fig. 2. Run Substitution Sample Paths.

For a given run threshold of K quats, the maximum run lengths after substitution for unslotted and slotted substitution are $K-1$ and $2K-2$ quats respectively. Assuming the worst case, as with killer packet attacks for example, the received symbol stream will appear as a maximum length run alternating with a substitution code. The substitution scheme and the value of K must be chosen such that clock recovery is not compromised. Maximum run lengths for different values of K are given in Table 2.

Table 2. Maximum run lengths for different values of K.

K	Maximum Run Length (Quats)	
	Slotted	Unslotted
4	6	3
8	14	7
16	30	15
24	46	23
32	62	31

Excluding killer packet attacks, the scrambled data stream will appear to be random if either the scrambler *or* the unscrambled data pattern is random. A well-designed scrambler should produce, for all practical purposes, a random bit stream. Likewise, the fields of Ethernet frames are generally random. The principal benefit then from scrambling comes from randomizing repeated patterns, such as the preamble, carrier extension and idle sequences. In the absence of killer packet attacks, it ought to suffice to study only the case where the scrambled stream is random. Table 3 shows the run length distribution without substitution. While a "run" usually means a constant symbol stream, the runs "seen" for clock recovery purposes may differ however, depending on implementation. A simplified edge detection circuit may only be detecting transitions between pairs of levels (e.g., (+3,-3) and (+1,-1)) for example. Accordingly, two sets of probabilities are shown. The probabilities for $p = 0.25$ correspond to constant symbol runs (applicable to clock recovery circuits that detect all transitions), while those for $p = 0.5$ correspond to runs consisting of any one of two symbol pairs. It can be seen that in either case, long runs are extremely improbable. For the case where substitution is used to bound run lengths, the corresponding distributions have also been evaluated to completely characterize the resulting truncated runs. Distributions for $K = 8$ are given in Tables 4 and 5 for slotted and unslotted substitution respectively. The numerical values were obtained from theoretical results derived in Appendix A. For the sake of brevity, only a partial listing of the probability distributions are shown, and corresponding results for other values of K have been omitted.

Table 3. Run length distribution without substitution.

Run Length	Probability	
	p = 0.25	p = 0.5
1	0.75	0.5
2	0.1875	0.25
4	0.01172	0.0625
8	4.578e-5	0.00391
16	6.985e-10	1.526e-5
32	1.626e-19	2.33e-10
64	8.816e-39	5.42e-20

Table 4. Run length distribution for slotted substitution, K=8.

Run Length	Probability	
	p = 0.25	p = 0.5
1	0.75	0.5
2	0.188	0.25
4	0.0117	0.0626
8	4.01×10^{-5}	0.00342
10	1.79×10^{-6}	0.000609
14	1.4×10^{-9}	7.61e-6

Table 5. Run length distribution for unslotted substitution, K=8.

Run Length	Probability	
	p = 0.25	p = 0.5
1	0.75	0.502
2	0.1875	0.251
4	0.01172	0.06275
7	1.831×10^{-4}	0.0311

It can be seen in Table 3 that the number of long runs (e.g., 32, or 64 quats) becomes statistically a very small fraction of the total number of runs. The corresponding proportion of time occupied by long runs is very small. As an example, it can be shown that the proportion of time occupied by runs longer than 64 quats for $p = 0.25$ is 1.44×10^{-37}, or roughly 1 run longer than 64 quats in 4.4×10^{20} years. SONET requires that clock recovery be possible for runs of up to 72 bits. PMC-Sierra's chips are typically capable of maintaining sync for runs of even up to 80 bits (line symbols) long. It is clear that in general, if killer packets are excluded, a good scrambler alone (without run substitution) should be sufficient to ensure reliable clock recovery.

IV. Recommendations for PMC's Gigabit Ethernet UTP-5 PHY Proposal.

Frame synchronous cipher stream scrambling is recommended to avoid error multiplication inherent with self-synchronous scrambling, and possible bandwidth expansion as well as additional implementation complexity resulting from distributed sample scrambling. The killer packet problem is addressed by a substitution scheme in which constant runs are replaced by uniquely identifiable patterns to restore the requisite transition density required for clock recovery [5]. Since constant runs also occur, however improbable (as analyzed below) in completely random, scrambled data, the substitution code is also of incidental utility to *guarantee* transition density, a characteristic of 8B10B-coded data transmission.

In some applications, error multiplication may not be a major issue, as a packet with one error is no better off than a packet with multiple errors. In the case of IEEE

802.3 however, there is a strong requirement that any combination of up to 3 bit errors in an Ethernet frame must be detectable. The possible existence of subsequent bit errors arising from any of up to 3 originating errors makes it difficult to prove that the error detection requirement can be satisfied, if at all. The error detection properties of the CRC-32 code used in the FCS would have to be characterized thoroughly and compared with patterns containing more than 3 bit errors. Moreover, the existence of even one undetectable pattern arising from error multiplication from 3 or fewer bit errors might well be enough to render the scrambling scheme unacceptable in an 802.3 standard. It is reasonable to conjecture that consideration of error multiplication precluded self-synchronous scrambling from the ANSI TP-PMD spec.

It may very well suffice to simply adopt the scrambler specified in ANSI TP-PMD. The frequency spectrum of the bit stream produced by this LFSR does appear to correspond to that of a random data stream [6]. In the PMC scheme, the scrambler is initialized at certain reference points marked by special unscrambled symbol patterns embedded in the transmitted data stream (e.g., start of packet, link unavailable indication and link configuration). The scrambled bits pass through a run detector and a substitution encoder. The run detector can be implemented by a bit serial FIFO with a digital comparator to detect bit patterns that will generate runs. The run detector's decisions feed into the substitution encoder, which either passes the quat stream from the FIFO or generates a run substitution code. At the receiver, the received symbols are buffered and decoded. Control codes are extracted and runs corresponding to substitution codes are replaced. The descrambler uses the same LFSR as used by the scrambler. It is initialized with the same seed at the same initialization points for scrambling, as determined by control codes extracted from the received symbol stream. A block diagram is shown in Fig. 3. Note that delays equal to the run substitution threshold are incurred at both the transmitter and the receiver. While a simpler implementation is possible with an edge detection circuit that only detects transitions between pairs of levels, the required number of substitution codes may increase enormously. This is because the number of distinct symbol sequences increases exponentially with run length, as shown below in Table 6. Consequently, the number of substitution codes required can be prohibitively large even for short runs. For this reason, an edge detection that can detect transitions between any symbol level is strongly recommended.

Table 6. Number of distinct symbol sequences for runs consisting of symbol pairs.

Run Length	Number of Distinct Runs
4	32
8	512
16	131072

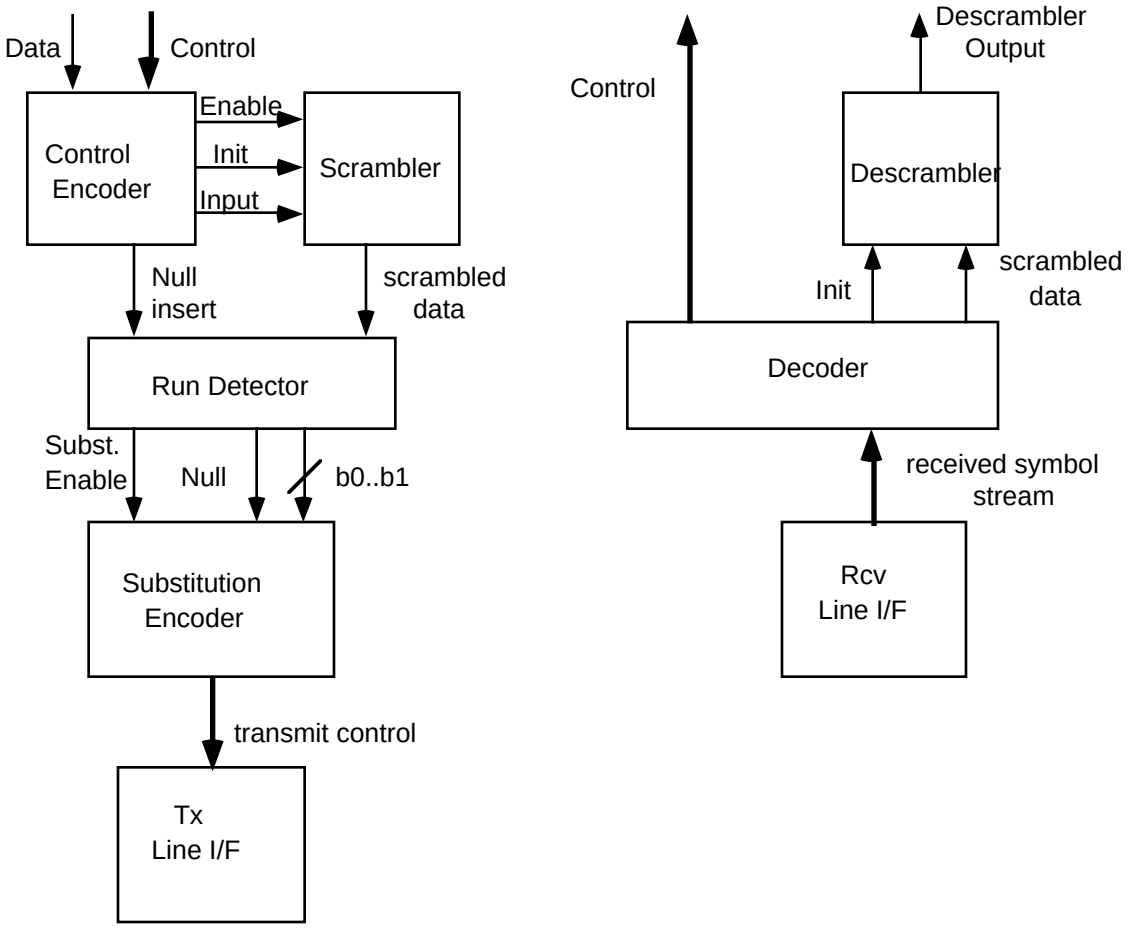

Fig. 3. Block diagram of scrambler and substitution functions.

V. Concluding Remarks

In summary, scrambling is recommended for EMC reasons. Frame synchronous cipher-stream scrambling is recommended over self-synchronous scrambling for robustness against error multiplication. Compared to distributed sample scrambling, frame synchronous scrambling avoids bandwidth expansion and sample error sensitivity issues. Moreover, distributed sample scrambling does not confer any significantly improved scrambling characteristic for the relatively long frame lengths encountered in Ethernet. Run substitution is proposed to remove sync loss susceptibility of frame-synchronous scrambling to killer packet attacks, and of scrambling in general to random occurrences of long constant runs. Detection of transitions between any of the four levels (+1, +3, -1, -3) is strongly recommended and may well be required in order to control the number (and possibly the complexity as well) of substitution codes. It is difficult to discuss how much more robust unslotted substitution is to slotted substitution without further details on the substitution codes as well as the channel model. The difference in robustness is

expected to be marginal. At any rate, slotted substitution with $K = 8$ appears to be a reasonable choice. The TP-PMD scrambler has a period of approximately 1000 quats. With a line rate of 250 MBd, this translates to a frequency of approximately 250 kHz, yielding spectral contributions that are expected to be far below the range for radiated emissions. No particular problems are foreseen at this point for conducted emissions either. While the TP-PMD scrambling scheme appears to be acceptable, other LFSR configurations may be suitable as well. For example, the generator polynomial for the distributed sample scrambler used for the ATM cell-based PHY ($1 + x^{28} + x^{31}$) might also be considered for frame synchronous scrambling.

VI. References

[1] American National Standards Institute, "Fibre Distributed Data Interface (FDDI) - Token Ring Twisted Pair Physical Layer Medium Dependent (TP-PMD)", ANSI X3.263-1995.

[2] Bellcore, "Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria", TA-NWT-000253, Issue 6, Sept. 1990.

[3] ITU-T Recommendation I.432, "B-ISDN User Network Interface - Physical Layer Specification", June 1992.

[4] De Prycker, M., "Asynchronous Transfer Mode: Solution for Broadband ISDN, 3rd ed.", UK: Prentice-Hall, 1995.

[5] Cam, R., ("4LZS with substitution?", internal e-mail correspondence, Sept. 23, 1996.)

[6] Gerson, B., (Internal communication. Preliminary results from simulation, Oct. 1, 1996.)

Appendix A. Run Length Analysis for Random Data Streams.

An unchanging sequence of symbols is generated when bit patterns repeat, taken two at a time. All the possible patterns are shown in Table 1. Let p denote the probability that the next two-bit sequence exactly matches the preceding two-bit sequence. Let $q = 1 - p$ denote the complementary probability. Since 1's and 0's occur with equal probability, $p = 1/4$. In the quasi-binary case (considering only transitions between pairs of levels, e.g., (+3,-3) and (+1,-1)), $p = 1/2$.

A.1. Run Length Distribution Without Substitution.

The behavior of the runs with time can be modeled by a Markov chain, from which the probability distribution for the run length can be obtained from the stationary distribution of the Markov chain. As depicted in Fig. A.1, the system is initially in state 1. Each time the current symbol is repeated (with probability p), the system moves to state 2, 3, 4, ... and so forth. If a different symbol occurs (with probability q), the system returns back to state 1 (thus ending a given run). The stationary probabilities can be obtained as follows. Let π_i denote the probability that the system is in state i, $i = 1,2,\ldots$ Using flow balance on nodes,

$$\begin{aligned} p\pi_1 &= (p+q)\pi_2 = \pi_2 \\ p\pi_2 &= \pi_3 = p^2\pi_1 \\ &\vdots \\ p\pi_{i-1} &= \pi_i = p^{i-1}\pi_1 \end{aligned} \tag{A.1}$$

Since $\sum_{i=1}^{\infty}\pi_i = 1$, $\pi_1(1+p+p^2+\cdots) = \pi_1/(1-p) = 1$. So $\pi_1 = q$, from which all the other probabilities can be computed.

The proportion of time, F, occupied by runs longer than K symbols is given by

$$\begin{aligned} F &= \frac{\sum_{i=K+1}^{\infty} i\pi_i}{\sum_{i=1}^{\infty} i\pi_i} \\ &= (K+1)p^K q + p^{K+1} \end{aligned} \tag{A.2}$$

after substituting in (A.1) and simplifying.

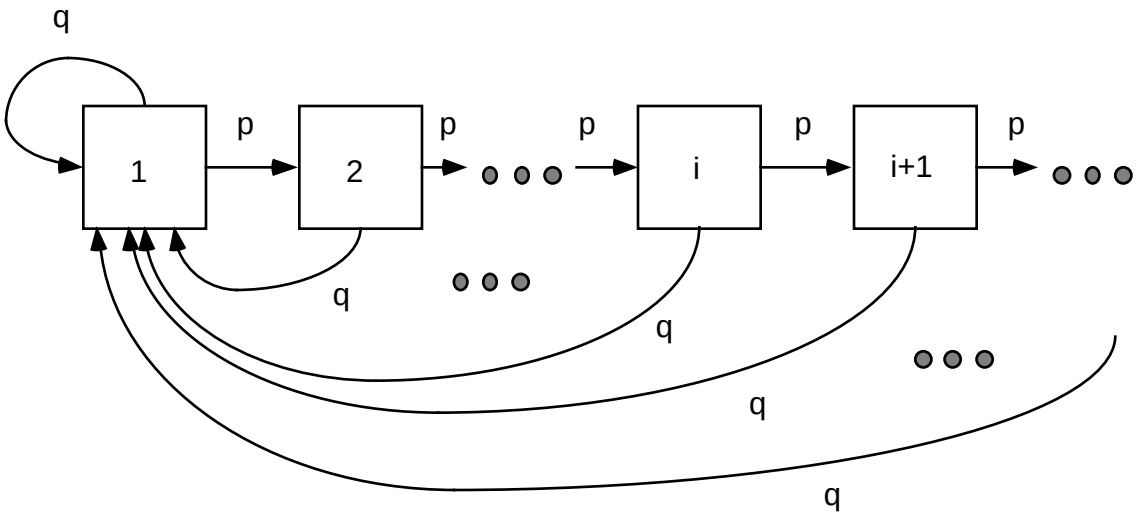

Figure A.1. Markov chain for run lengths without substitution.

A.2. Run Length Distribution with Unslotted Substitution.

With unslotted substitution, any run K symbols long is substituted with a pattern of equal length. The runs "seen" on the transmitted data will therefore consist of unsubstituted runs (shorter than K-1 symbols), and substitution codes replacing runs K symbols long. The choice of the substitution codes can therefore affect the resulting runs in the (substituted) data stream to some extent. To avoid complications arising from dependencies on the substitution code and for the sake of clarity, a run preceding a substitution is considered to be terminated (ended) at the symbol immediately preceding the substitution codeword. Runs containing the substitution code are not considered. The impact of this approximation is not serious, as the worst-case runs can be easily bounded, by adding in the length of the substitution code for example.

A Markov chain similar to that used for the case without substitution can be used as follows. The run lengths (with the definition above in mind) are mapped onto states in the Markov chain. As shown in Fig. A.2, transitions occur from state *i* to *i+1* with probability *p*, and from any state *j* to *1* with probability *q*, except for state *K*, wherein a run of length *K* occurs and is substituted. State *K* transitions to state *1* with probability *1*. Using the flow balance equations,

$$\pi_i = p^{i-1}\pi_1, i = 2,\ldots,K. \tag{A.3}$$

Since $\sum_{i=1}^{K}\pi_i = 1$,

$$\pi_1 = \frac{1-p}{1-p^K}. \tag{A.4}$$

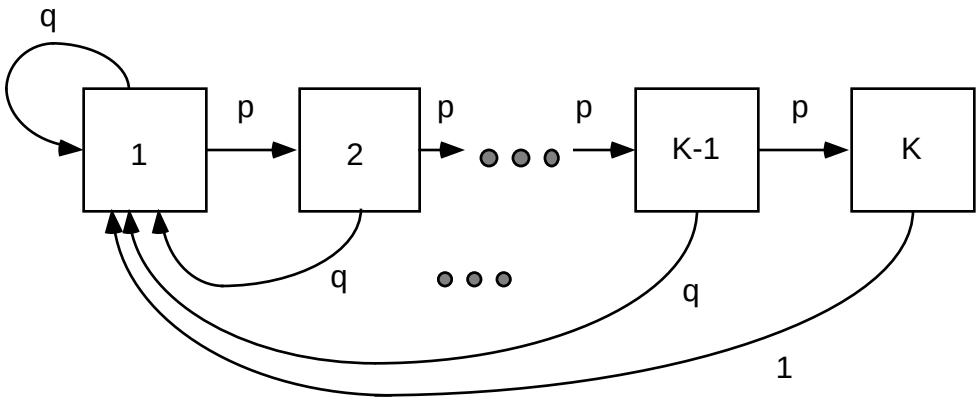

Fig. A.2. Markov chain for run lengths with unslotted substitution.

A.3. Run Length Distribution with Slotted Substitution.

The analysis here is somewhat more complicated because substitution at slotted intervals introduces memory in the system. The maximum run length (before substitution is triggered) will depend on where the run starts relative to the slot boundary. A *K*-state Markov chain can be constructed to model starting points of the runs. The run length distribution can then be obtained by conditioning on the distribution of the starting points.

In general, the $(K \times K)$ state transition matrix takes the following form:

$$P = \begin{pmatrix} 0 & q & pq & p^2q & \cdots & p^{K-3}q & p^{K-2} \\ p^{K-2}q & p^{K-1}q & q+p^Kq & pq & \cdots & p^{K-4}q & p^{K-3}q+p^{K+1} \\ \vdots & & & & \ddots & & \vdots \\ pq & p^2q & p^3q & p^4q & \cdots & p^{K-1}q & q+p^K \\ q & pq & p^2q & p^3q & \cdots & p^{K-2}q & p^{K-1} \end{pmatrix}. \qquad \text{(A.5)}$$

To see how this matrix is constructed, consider the first row. This corresponds to the first position in the slot. With probability q, the run (of length 1) is ended, and the next starting point is at position 2. With probability pq, a symbol is repeated, followed by a different symbol (ending a 2-symbol run), and the next starting point is at position 3. When *K*-2 repeats occur (with probability p^{K-2}), the run will be terminated if the next symbol is either different or repeated (because a substitution is inserted to replace the *K* symbols). Examples for *K*=3 and *K*=4 are given by

$$P = \begin{pmatrix} 0 & q & p \\ pq & p^2q & q+p^3 \\ q & pq & p^2 \end{pmatrix} \tag{A.6}$$

and

$$P = \begin{pmatrix} 0 & q & pq & p^2 \\ p^2q & p^3q & q+p^4q & pq+p^5 \\ pq & p^2q & p^3q & q+p^4 \\ q & pq & p^2q & p^3 \end{pmatrix} \tag{A.7}$$

respectively.

In general, the stationary distribution is obtained by solving the system of linear equations given by

$$\sum_{j=1}^{K} P_{ji}\pi_j = \pi_i, i = 1,\ldots,K, \tag{A.8}$$

subject to

$$\textstyle\sum_{i=1}^{K} \pi_i = 1. \tag{A.9}$$

With this particular problem, a simple solution by direct application of flow balance either on nodes or on arcs did not appear to be possible. By solving for $K = 3$ and $K = 4$ however, a pattern emerged, indicating that $\pi_1 = \pi_2 = \cdots = \pi_{K-1} = \gamma$, for some constant γ that changed with K. By conjecturing that this was true, the value of γ was obtained as follows. Applying (A.9),

$$(K-1)\gamma + \pi_K = 1. \tag{A.10}$$

Let $S_i \equiv \sum_{j=1}^{K-1} P_{ji}$. From (A.8), we have

$$S_K\gamma + P_{KK}\pi_K = \pi_K. \tag{A.11}$$

By combining (A.10) and (A.11), and solving for γ by eliminating π_K, we obtain

$$\gamma = \frac{1-P_{KK}}{S_K + (K-1)(1-P_{KK})}. \tag{A.12}$$

π_K can then be obtained from (A.10). The conjecture was tested and found to be true at least for the numerical results in this report.

Let L and I denote the run length and starting point respectively. The run length distribution can be computed by conditioning on the π_i's as follows,

$$\Pr(L = x) = \sum_{i=1}^{K} \Pr\big(L = x | I = i\big)\pi_i, \tag{A.13}$$

where

$$\Pr(L = x | I = i) = \begin{cases} p^{x-1}q, x = 1,\ldots,2K - i, x \neq k - i + 1, i = 2,\ldots,K \\ p^{x-1}q + p^{2K-i}, x = K - i + 1, i = 2,\ldots,K \\ p^{x-1}q, x = 1,\ldots,K. \end{cases} \tag{A.14}$$