

Nonce in TCP Quick Start (draft)

Guohan Lu

Dept. of Electronic Engineering

Tsinghua University

Beijing, P.R.China

lguohan@gmail.com

September 23, 2005

1 Introduction

The current TCP quick start draft [1] proposes to use nonce to prevent receivers from lying. However, it doesn't provide protection against receivers reporting that the Rate Request was decremented by only one step. In this article, we propose another way of constructing the nonce that provides level-by-level protection. The nice property of our method is that the more the receiver wants to raise the rate, the less probability it can succeed.

2 Providing Level-by-level Protection

The main idea of our method is to divide the Request Rates into several levels and use one bit in the nonce to protect one level. For example, as showed in Table 1, we can divide the 16 Request Rates (0 ~ 15) into five levels. For each of the first 4 levels, we assign one bit in the nonce to it.

Same as the draft, the four-bit nonce is initialized by the sender to a random value. However, when the router reduces the Rate Request field, it does not zero all the bits in the nonce. It only zeros those which correspond to the levels higher than that of the reduced

level	Request Rates	bit position in the nonce
5	15,14,13	1
4	12,11,10,9	2
3	8,7,6,5	3
2	4,3,2,1	4
1	0	null

Table 1: 5 levels

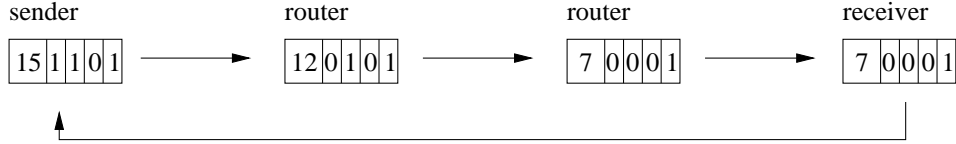


Figure 1: An example of nonce usage in a request

rate. For example, when the rate is reduced from 15 (level 5) to 7 (level 3), the 1st and 2nd bits in the nonce are zeroed. When the rate is reduced from 7 to 6, no bit in the nonce is zeroed.

When the Request Rate and the Nonce are reported back from the receiver, the bits in nonce which protect the reported rate must match exactly with the original sent value. Otherwise, it's an invalid response. For example, when the reported rate is 15, 14 or 13, all four bits in the Nonce must be matched. When the rate is 8, 7, 6 or 5, only the 3rd and 4th bits in the Nonce need to be matched.

For a lying receiver, when it wants to raise the rate to one level higher, it needs to guess the original bit in the nonce that protects that level. Thus, it has 50-50 chance to succeed. When it wants to raise the rate to two levels higher, it has 1/4 chance to succeed. The more it wants to raise, the less probability it can succeed. In Fig. 1, if the receiver wants to increase rate 7 to rate 15, it has to guess the first two bits (11) in the original nonce, that's 1/4 chance. By using different bits in the nonce to protect different rate levels, this method can detect a lying receiver when it tries to raise the rate to a higher level.

However, there are two main drawbacks of this method:

1. No protection for the rates in the same level. The lying receiver can always succeed in reporting the largest rate in the level they receive, as long as the largest rate is less or equal than the rate initiated by the sender.
2. Wasted bits in the nonce. When the Request Rate initiated by the sender is below level 5, the first bit in the nonce is wasted because it does not protect anything.

In § 3, we give an improved method by playing some trade-offs. However, we note that the first drawback cannot be eliminated because there are only 4 bits in the nonce. If we can have 15 bits in the nonce, we can protect each rate with one bit. Then, we can detect a lying receiver even if it only tries to raise the rate one step higher. If it wants to raise the rate from 1 to 15, it needs to guess 14 missing bits in the nonce, i.e. 2^{-14} chance to succeed. However, it means that we need to increase the length of the option with two more bytes.

3 Four-bits Floating Nonce

To overcome the previous two drawbacks, we can

1. decrease the number of rates each level contains,

bit pos.	rates
1	12, 11, 10
2	9, 8, 7
3	6, 5, 4
4	3, 2, 1

Table 2: 4 floating levels

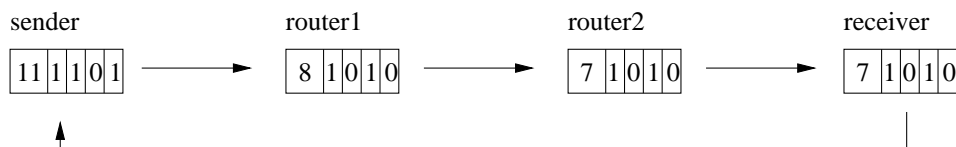


Figure 2: Left shifting bits in the nonce when decreasing the request rate

2. make the first bit in the nonce always correspond to the level of the current Request Rate field. Because the bits in the nonce do not correspond to fixed levels, we call this a floating nonce.

However, as we still only have 4-bits in the nonce, the benefits do not come with no price. It leaves some small rates unprotected as we'll illustrate later in this section.

We classify 1 ~ 15 rates into 5 levels, each level contains three rates. Rates 15, 14, 13 are in the 5th level, rates 12, 11, 10 are in the 4th level, and so on. Assume R is the current Requested Rate, the 1st bit in the nonce corresponds to level $l = \text{ceil}(R/3)$, the 2nd bit corresponds to the level one level lower, and so on. Table 2 shows the rates protected by the nonce when $R = 11$.

In this method, when the router decreases the Request Rate from level n to level $n - i$, it then left shifts the nonce i bits (See Fig. 2 for example). The router1 decreases the rate one level lower, so it left shift the nonce 1 bit. The router2 decreases the rate, but in the same level, so it doesn't left shift the nonce.

This method has two advantages over the previous one:

1. Fewer rates in the level. There are only three rates in each level, the receiver can raise at most two steps higher without guessing the missing the nonce.
2. Fewer wasted bits in the nonce. For the previous method, there is a wasted bit in the nonce when the requested rate is below 13. For this method, it happens when the requested rate is below 10.

The drawback of the method is that when the Request Rate initiated by the sender is in level 5, then the rates in level 1 (i.e 3, 2, 1) leave unprotected as there is no room left to associate one-bit nonce with this level. A lying receiver can report rate 3 safely if he knows the original Request Rate is in level 5. Although partitioning rates into even more levels further decreases inner-level lying, it'll leave more rates unprotected when the original Request Rate is large.

4 Conclusion

In this article, we showed an alternative way to use the 4 reserved bits as Rate-Reduce Nonce. Compared to the method proposed in the draft, our method provides level-by-level protection when the receiver tries to raise the received request rate. The more it tries to raise, the less probability it can succeed. However, with the limitation of 4-bits, there may be 2 or 3 rates in each level, receivers can have inner-level cheat without being detected.

If we can have a 15-bit nonce, we can protect every rate with one bit nonce. A lying receiver can be detected even it tries to raise the rate one step higher. Its winning probability decreases to 2^{-n} when it tries to raise the rate n step higher.

References

- [1] A. Jain, S. Floyd, M. Allman, and P. Sarolahti. Quick-start for tcp and ip. draft-ietf-tsvwg-quickstart-00.txt.