

# Understanding the Definition of bits per second (bits/sec) from the show interfaces Command Output

Document ID: 12816

---

<b>Introduction</b>
<b>Prerequisites</b>
Requirements
Components Used
Conventions
<b>Definition of bits per second</b>
<b>Related Information</b>

---

## Introduction

This document answers the question "What is the definition of bits/sec in the output of the **show interfaces** command?"

## Prerequisites

### Requirements

There are no specific requirements for this document.

### Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

### Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

## Definition of bits per second

Bits per second includes all packet/frame overhead. It does not include stuffed zeros. The size of each frame is added to the total bytes of output. Take the difference every 5 seconds to calculate the rate.

The algorithm for the five-minute moving average is:

$$\text{new average} = ((\text{average} - \text{interval}) * \exp(-t/C)) + \text{interval}$$

where:

- t is five seconds, and C is five minutes.  $\exp(-5/(60*5)) == .983$ .

- newaverage = the value we are trying to compute.
- average = the "newaverage" value calculated from the previous sample.
- interval = the value of the current sample.
- (.983) is the weighting factor.

Here, you take the average from the last sample, less what was gathered in this sample, and weight that down by a decay factor. This quantity is referred to as a "historical average". To the weighted (decayed) historical average, add the current sample, and come up with a new weighted (decayed) average.

The interval is the value for some given variable in the five-second sample interval. The interval can be load, reliability, or packets per second. These are the three values to which we apply exponential decay.

The average value minus the current value is the deviation of the sample from the average. You must weight this by .983, and add it to the current value.

If the current value is greater than the average, this results in a negative number, and causes the "average" value to rise less quickly on traffic spikes.

Conversely, if the current value is less than the running average, it results in a positive number, and ensures that the "average" value falls less rapidly if there is a sudden stoppage of traffic.

Imagine that traffic is stopped altogether, after it has been 100% for an infinite period before such stoppage. In other words, the average rose slowly to 100%, and stayed there. The interval is always 0 for the "no traffic" scenario. Then, over five-second intervals, the exponentially weighted utilization goes from:

$$1.0 - .983 - .983^2 - .983^3 - \dots - .983^n$$

or

$$1.0 - .983 - .95 - 0.9 - 0.86 -$$

and so on.

In this example, utilization drops from 100% to 1% in 90 intervals, or 450 seconds, or 7.5 minutes. Conversely, if you start from 0 load, and apply 100% load, the exponentially decayed average should take about 7.5 minutes to reach 99%.

As  $n$  gets large (with time), the average slowly falls (asymptotically) to zero for no traffic, or climbs to 100% for maximum traffic.

This method prevents traffic spikes from skewing statistics about the "average". We are "damping" the wild fluctuations of the network traffic.

In the real world, where things are not so black and white, the exponentially decayed average gives a picture of your average network utilization untainted by wild spikes.

---

## Related Information

- [Technical Support – Cisco Systems](#)
-

