If you're new to this series, you can find the complete list of articles in the series on my personal page here.

If you are writing applications to sit behind a Load Balancer, it behooves you to at least have a clue what the algorithm your load balancer uses is about. We’re taking this week’s installment to just chat about the most common algorithms and give a plain-programmer description of how they work. While historically the algorithm chosen is both beyond the developers’ control, you’re the one that has to deal with performance problems, so you should know what is happening in the application’s ecosystem, not just in the application. Anything that can slow your application down or introduce errors is something worth having reviewed.

For algorithms supported by the BIG-IP, the text here is paraphrased/modified versions of the help text associated with the Pool Member tab of the BIG-IP UI. If they wrote a good description and all I needed to do was programmer-ize it, then I used it. For algorithms not supported by the BIG-IP I wrote from scratch. Note that there are many, many more algorithms out there, but as you read through here you’ll see why these (or minor variants of them) are the ones you’ll see the most.

Plain Programmer Description: Is not intended to say anything about the way any particular dev team at F5 or any other company writes these algorithms, they’re just an attempt to put the process into terms that are easier for someone with a programming background to understand. Hopefully a successful attempt.

Interestingly enough, I’ve pared down what BIG-IP supports to a subset. That means that F5 employees and aficionados will be going “But you didn’t mention…!” and non-F5 employees will likely say “But there’s the Chi-Squared Algorithm…!” (no, chi-squared is theoretical distribution method I know of because it was presented as a proof for testing the randomness of a 20 sided die, ages ago in Dragon Magazine). The point being that I tried to stick to a group that builds on each other in some connected fashion. So send me hate mail… I’m good. Unless you can say more than 2-5% of the world’s load balancers are running the algorithm, I won’t consider that I missed something important. The point is to give developers and software architects a familiarity with core algorithms, not to build the worlds most complete lexicon of algorithms.

- **Random:** This load balancing method randomly distributes load across the servers available, picking one via random number generation and sending the current connection to it. While it is available on many load balancing products, its usefulness is questionable except where uptime is concerned – and then only if you detect down machines.

  Plain Programmer Description: The system builds an array of Servers being load balanced, and uses the random number generator to determine who gets the next connection… Far from an elegant solution, and most often found in large software packages that have thrown load balancing in as a feature.

- **Round Robin:** Round Robin passes each new connection request to the next server in line, eventually distributing connections evenly across the array of machines being load balanced. Round Robin works well in most configurations, but could be better if the equipment that you are load balancing is not roughly equal in processing speed, connection speed, and/or memory.
**Plain Programmer Description:** The system builds a standard circular queue and walks through it, sending one request to each machine before getting to the start of the queue and doing it again. While I’ve never seen the code (or actual load balancer code for any of these for that matter), we’ve all written this queue with the modulus function before. In school if nowhere else.

- **Weighted Round Robin** (called **Ratio** on the BIG-IP): With this method, the number of connections that each machine receives over time is proportionate to a ratio weight you define for each machine. This is an improvement over Round Robin because you can say “Machine 3 can handle 2x the load of machines 1 and 2”, and the load balancer will send two requests to machine #3 for each request to the others.

**Plain Programmer Description:** The simplest way to explain for this one is that the system makes multiple entries in the Round Robin circular queue for servers with larger ratios. So if you set ratios at 3:2:1:1 for your four servers, that’s what the queue would look like – 3 entries for the first server, two for the second, one each for the third and fourth. In this version, the weights are set when the load balancing is configured for your application and never change, so the system will just keep looping through that circular queue. Different vendors use different weighting systems – whole numbers, decimals that must total 1.0 (100%), etc. but this is an implementation detail, they all end up in a circular queue style layout with more entries for larger ratings.

- **Dynamic Round Robin** (Called **Dynamic Ratio** on the BIG-IP): is similar to Weighted Round Robin, however, weights are based on continuous monitoring of the servers and are therefore continually changing. This is a dynamic load balancing method, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time. This **Application Delivery Controller** method is **rarely** available in a simple load balancer.

**Plain Programmer Description:** If you think of Weighted Round Robin where the circular queue is rebuilt with new (dynamic) weights whenever it has been fully traversed, you’ll be dead-on.

- **Fastest:** The Fastest method passes a new connection based on the fastest response time of all servers. This method may be particularly useful in environments where servers are distributed across different logical networks. On the BIG-IP, only servers that are active will be selected.

**Plain Programmer Description:** The load balancer looks at the response time of each attached server and chooses the one with the best response time. This is pretty straight-forward, but can lead to congestion because response time **right now** won’t necessarily be response time in 1 second or two seconds. Since connections are generally going through the load balancer, this algorithm is a lot easier to implement than you might think, as long as the numbers are kept up to date whenever a response comes through.

These next three I use the BIG-IP name for. They are variants of a generalized algorithm sometimes called Long Term Resource Monitoring.

- **Least Connections:** With this method, the system passes a new connection to the server that has the least number of current connections. Least Connections methods work best in environments where the servers or other equipment you are load balancing have similar capabilities. This is a dynamic load balancing method, distributing connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time. This **Application Delivery Controller** method is **rarely** available in a simple load balancer.

**Plain Programmer Description:** This algorithm just keeps track of the number of connections attached to each server, and selects the one with the smallest number to receive the connection. Like fastest, this can cause congestion when the connections are all of different durations – like if one is loading a plain HTML page and another is running a JSP with a ton of database lookups. Connection counting just doesn’t account for that scenario very well.
• **Observed**: The Observed method uses a combination of the logic used in the Least Connections and Fastest algorithms to load balance connections to servers being load-balanced. With this method, servers are ranked based on a combination of the number of current connections and the response time. Servers that have a better balance of fewest connections and fastest response time receive a greater proportion of the connections. This Application Delivery Controller method is rarely available in a simple load balancer.

Plain Programmer Description: This algorithm tries to merge Fastest and Least Connections, which does make it more appealing than either one of the above than alone. In this case, an array is built with the information indicated (how weighting is done will vary, and I don’t know even for F5, let alone our competitors), and the element with the highest value is chosen to receive the connection. This somewhat counters the weaknesses of both of the original algorithms, but does not account for when a server is about to be overloaded – like when three requests to that query-heavy JSP have just been submitted, but not yet hit the heavy work.

• **Predictive**: The Predictive method uses the ranking method used by the Observed method, however, with the Predictive method, the system analyzes the trend of the ranking over time, determining whether a server's performance is currently improving or declining. The servers in the specified pool with better performance rankings that are currently improving, rather than declining, receive a higher proportion of the connections. The Predictive methods work well in any environment. This Application Delivery Controller method is rarely available in a simple load balancer.

Plain Programmer Description: This method attempts to fix the one problem with Observed by watching what is happening with the server. If its response time has started going down, it is less likely to receive the packet. Again, no idea what the weightings are, but an array is built and the most desirable is chosen.

You can see with some of these algorithms that persistent connections would cause problems. Like Round Robin, if the connections persist to a server for as long as the user session is working, some servers will build a backlog of persistent connections that slow their response time. The Long Term Resource Monitoring algorithms are the best choice if you have a significant number of persistent connections. Fastest works okay in this scenario also if you don’t have access to any of the dynamic solutions.

That’s it for this week, next week we’ll start talking specifically about Application Delivery Controllers and what they offer – which is a whole lot – that can help your application in a variety of ways.
Until then!

Don.