This article describes the WAAS architecture and how data flows into, gets processed, and flows out of a WAAS device. It provides a basic understanding of these concepts to assist you in troubleshooting the WAAS system.

Guide Contents
Main Article
Understanding the WAAS Architecture and Traffic Flow
Preliminary WAAS Troubleshooting
Troubleshooting Optimization
Troubleshooting Application Acceleration
Troubleshooting the CIFS AO
Troubleshooting the HTTP AO
Troubleshooting the EPM AO
Troubleshooting the MAPI AO
Troubleshooting the NFS AO
Troubleshooting the SSL AO
Troubleshooting the Video AO
Troubleshooting the Generic AO
Troubleshooting Overload Conditions
Troubleshooting WCCP
Troubleshooting AppNav
Troubleshooting Disk and Hardware Problems
Troubleshooting Serial Inline Clusters
Troubleshooting vWAAS
Troubleshooting WAAS Express
Troubleshooting NAM Integration

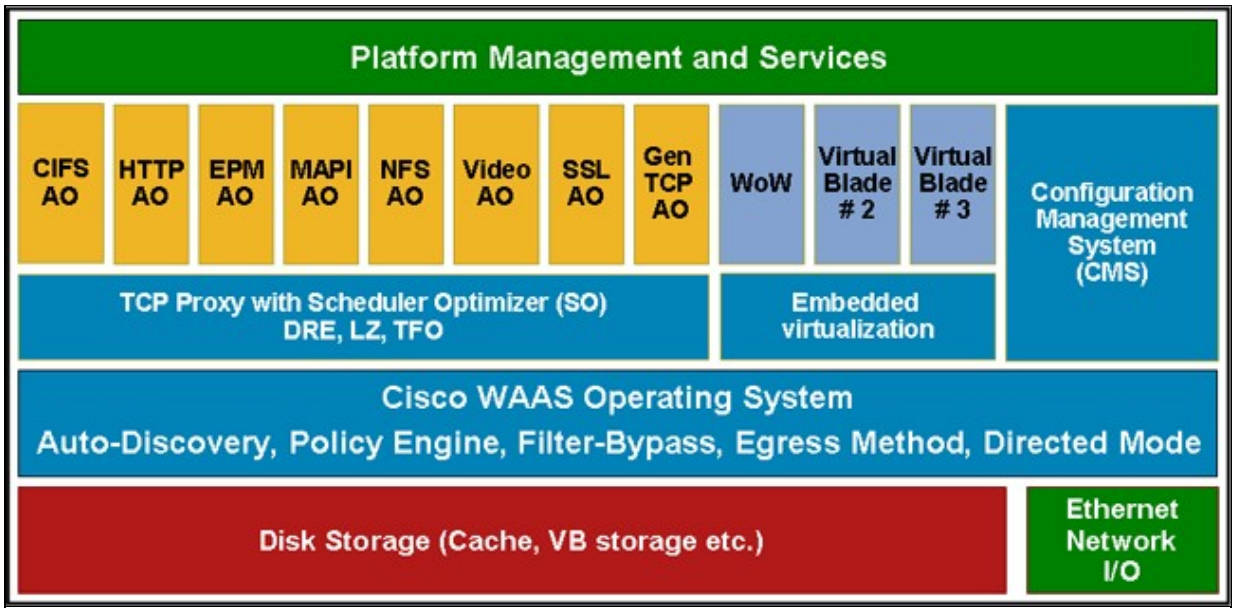
Contents

Understanding the WAAS Architecture

Having a basic understanding of the Wide Area Applications Services (WAAS) architecture and data flow can help to make troubleshooting the WAAS system easier. This section describes the major functional areas of the WAAS system and how they work together.

The WAAS system architecture is divided into a series of functional areas or services as shown in Figure 1.

Figure 1. WAAS Architecture



AOs

AOs (Application Optimizers, also known as Application Accelerators) are the application-specific software that optimizes certain protocols at Layer 7 (beyond the generic Layer 4 optimizations). AOs may be viewed as the "applications" in the WAE system (in an OS analogy). The generic AO acts as a catch-all for all traffic that has no protocol-specific AO and also functions as a delegate if a protocol-specific AO decides to not apply optimization.

WoW and Virtual Blades

Windows server on WAAS (WoW) is Microsoft Windows Server running in a virtual blade. The virtualization feature of WAAS allows you to configure one or more virtual blades, which are computer emulators that reside in a WAE or WAVE device. A virtual blade allows you to allocate WAE system resources for use by additional operating systems that you install on the WAE hardware. You can host third-party applications in the isolated environment provided by a virtual blade. For example, you could configure a virtual blade in a WAE device to run Windows printing and domain lookup services.

Configuration Management System

The configuration management system (CMS) consists of the WAAS Central Manager and its database for storing WAAS device configuration information. The CMS allows you to configure and manage WAE devices and device groups from a single Central Manager GUI interface.

DRE with Scheduler

The DRE with scheduler (SO-DRE) is the key module in the Layer 4 optimization space, and is responsible for all data-reduction techniques in the system, including data redundancy elimination (DRE) and persistent LZ compression. In addition to the system-wide algorithms for data reduction that are implemented here, this component also includes a scheduling element that allows the system to better control the order and pace of using DRE for different AOs.

Storage

The storage system manages the system disks and the logical RAID volumes on systems that have multiple disks. Disk storage is used for system software, the DRE cache, the CIFS cache, and virtual blade storage.

Network I/O

The network input/output component is responsible for all aspects that are related to handling data communication coming into or going out from a WAE, including WAE to WAE communication and WAE to client/server communication.

Interception & Flow Management

Interception and flow management consists of multiple submodules that, using policies configured by the user, intercept traffic, automatically discover peers, and start optimization on a TCP connection. Some of the key submodules are Auto-Discovery, Policy Engine, and Filter Bypass.

Auto-Discovery

Auto-discovery allows peer devices to discover each other dynamically and does not require that you preconfigure WAE pairs. Auto-discovery is a multi-WAE end-to-end mechanism that defines a protocol between the WAEs that discovers a pair of peer WAEs for a given connection.

WAE devices automatically discover one another during the TCP three-way-handshake that occurs when two nodes establish a TCP connection. This discovery is accomplished by adding a small amount of data to the TCP options field (0x21) in the SYN, SYN/ACK, and ACK messages. This TCP option allows WAE devices to understand which WAE is on the other end of the link and allows the two to describe which optimization policies they would like to employ for the flow. If intermediate WAEs exist in the network path, they simply pass through flows that are being optimized by other WAEs. At the end of the auto-discovery process, the WAEs shift the sequence numbers in the TCP packets between the participating WAEs by increasing them to more than 2 billion, to mark the optimized segment of the connection.

Policy Engine

The policy engine module determines if traffic needs to be optimized, which AO to direct it to, and the level of data-reduction (DRE) if any, that should be applied to it. The policy engine classifies traffic beyond connection establishment (for example, based on payload information) and changes the flow of a connection dynamically from unoptimized to optimized.

The elements of a policy include the following:

- Application definition: A logical grouping of traffic to help report statistics on the type of traffic.
- Traffic classifier: Access control list (ACL) that helps choose connections based on IP addresses, ports, and so on.
- Policy Map: Binds the application and the classifier with an action, which specifies the type of optimization, if any, to be applied. There are two kinds of policy maps:
    - ♦ Static policy map: Configured on the device through the CLI or GUI (or installed by default) and is persistent unless removed.
    - ♦ Dynamic policy map: Automatically configured by the WAE and has a life span just long enough to accept a new connection.

The following configuration example shows a policy engine application definition (Web) that includes a classifier (HTTP) and an action (optimize full accelerate http):

```
wae(config)# policy-engine application map basic
wae(config-app-bsc)# name Web classifier HTTP action optimize full accelerate http set-dscp copy
```

Filter-Bypass

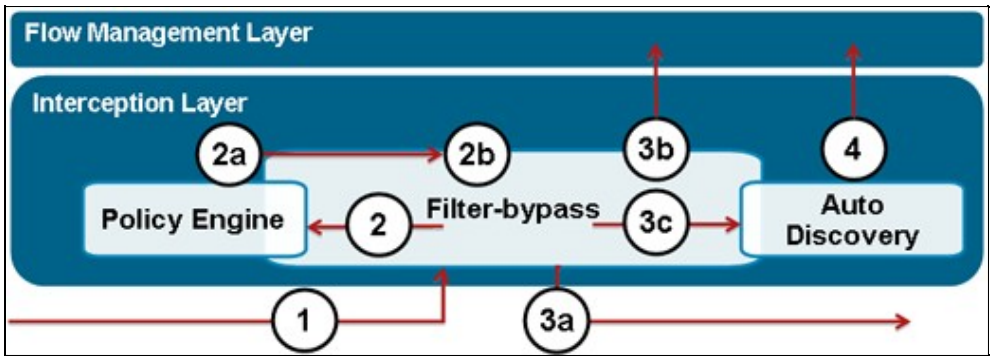
After interception, the filter-bypass module acts as the mediator between the policy engine and auto discovery. The filter-bypass module tracks all optimized connections in a filtering table for the life of the connection. Additionally, it tracks pass-through connections, but pass-through table entries are timed out after 3 seconds.

WAAS Traffic Flow

This section describes the packet flow in WAAS.

Figure 2 shows the filter-bypass flow establishment as a packet enters the system.

Figure 2. Filter-Bypass Flow Establishment



1. A SYN packet on a flow enters the system. This packet is routed to the filter-bypass module.

2. The filter-bypass module consults the policy engine on how the flow should be handled.

2a. The policy engine consults the configured and dynamically added policies, and based on the current operational status of the AOs and SO-DRE, decides what the WAE could do for this flow: pass through, locally terminate, or optimize.

2b. The packet and the decision from the policy engine are then returned to the filter-bypass module.

3. The filter-bypass module acts on the policy engine decision in one of the following ways:

3a. Sends the packet out immediately (pass through).

3b. Sends the packet up for local termination by an AO.

3c. Sends the packet to the auto-discovery module for optimization.

If the filter-bypass module chooses option 3c, the packet is sent to the auto-discovery module. The auto-discovery module determines what optimizations can be done, based on the availability of a peer WAE and its enabled features. A peer WAE is discovered through the use of TCP options added during the TCP handshake to the remote node. If the auto-discovery module determines that a peer WAE is available, the connection is handed off for further processing once the TCP three-way-handshake completes. If a peer WAE is discovered for the first time, the WAEs additionally negotiate about AO versions and capabilities. This information is used to decide the AO level capabilities for the connection.

4. The connection is finally admitted into the system with specific L4 and L7 optimizations and handed off to the appropriate L4 (DRE) and L7 (AO) acceleration modules. For connections that are later discovered to be not optimize-able by protocol-specific AOs (HTTP, MAPI, and so on), the connection is handled by the generic AO, with or without DRE optimization (as negotiated during connection establishment).