



Offloading Secure Connection Processing with AppBeat™ DC

Executive Summary

Security is essential in today's world as web applications transmit more and more sensitive information over the internet between clients and servers. However, encrypting and decrypting traffic for each client consumes a huge amount of bandwidth on a server, prohibiting it from performing other essential operations quickly and thus increasing user response time. Crescendo Networks' AppBeat DC solves this problem by handling all secure transaction processing, significantly reducing server overhead while still maintaining an application's security and privacy policies. By fully offloading SSL tasks from the entire server farm, AppBeat DC saves significant processing resources, allowing the server farm to continue to host the application optimally and with minimal overhead.

Overview

Crescendo Networks' AppBeat DC provides high-performance application delivery and acceleration for enterprises and web sites. AppBeat DC leverages the Maestro platform, bringing instant performance relief to overburdened data centers today. AppBeat DC is the only solution built to be an integral component in emerging data center architectures.

AppBeat DC offloads task-intensive functions from servers in a web application, optimizing that application and allowing the servers that host it to scale significantly. Deployed as an appliance, it front-ends the servers, intercepting and processing all user requests destined for them. By performing this function, AppBeat DC provides various optimization services for the servers in the application, massively improving server performance while reducing user response time and consumed bandwidth.

Secure Communications in Web Applications

Security is a vital component in today's web applications as more and more sensitive information is transmitted between clients and servers. This is done using the Secure Sockets Layer (SSL) protocol. Through comprehensive security algorithms, SSL can ensure privacy by encrypting traffic between clients and servers.

A single SSL session operates in two phases: asymmetric and symmetric. When a client first establishes a connection to an SSL-enabled server, a handshake process is performed using public

key cryptography (asymmetric encryption) to verify the identity of the server (and, optionally, the client). Certificates signed by publicly trusted entities are employed to facilitate this process. After this SSL handshake is completed, a set of shared keys is negotiated between the two endpoints and the actual transfer of bulk data begins via symmetric encryption schemes that use the just-negotiated keys.

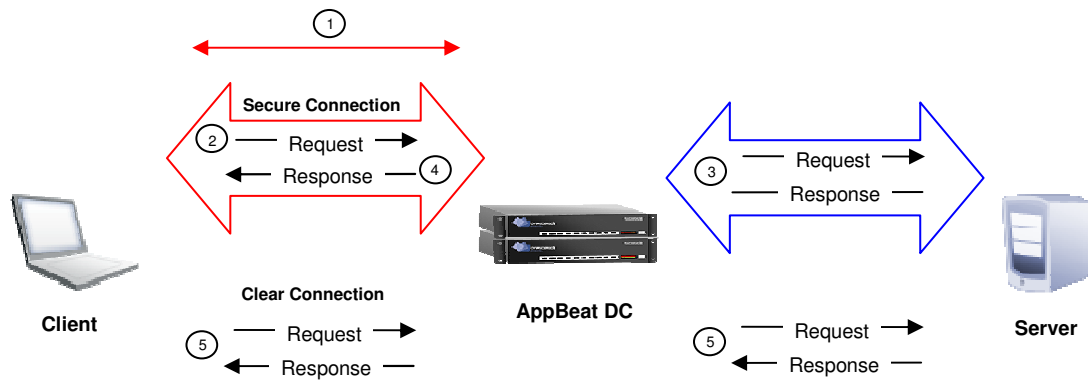
The problem with SSL communications is its extreme use of a server's processing resources. Public key cryptography is a major burden on a server's CPU, especially when it has to deal with a large number of clients trying to negotiate security schemes. This is compounded by the fact that relatively large cryptographic keys must be used when dealing with a public network, such as the Internet, to prevent information hijacking by malicious users. Typical servers can deal with SSL connections only at a small fraction of what they can handle in non-secure traffic.

Although the asymmetric phase of an SSL session presents the major burden to a server, the symmetric encryption used in actual bulk data transfer is also a pain point. Encryption/decryption must be performed by the server for every object carried over a secure session. This presents an additional amount of overhead for the server when dealing with secure connections.

AppBeat DC's SSL Offload

AppBeat DC provides SSL acceleration and offload modules. When fronting a server farm, AppBeat DC is capable of handling all secure transaction processing, significantly reducing the server overhead while still maintaining an application's security and privacy policies. The server no longer has to deal with the large number of secure connection setups or the bulk encryption/decryption performed on the data traversing those connections, allowing it to focus its resources on the application itself.

AppBeat DC's SSL functionality is enabled by a dedicated module within the Maestro platform that performs all phases of secure communications purely in hardware, including secure session setup and bulk data transfer. The module operates independently from the rest of the system, through dedicated hardware and memory, allowing AppBeat DC to scale significantly in both phases of SSL communications: session setup and bulk data encryption. Public and private keys, along with the associated certificates can be imported to the device as it takes over the responsibility for secure transactions. AppBeat DC is also capable of generating self-signed certificates. Full control is also provided over the various ciphers to be used for the secure sessions, along with their relative priorities. Communication to the servers continues over the existing optimized TCP/HTTP sessions that AppBeat DC has with each server. The following diagram shows the operational flow of secure connections being accelerated and optimized by AppBeat DC:



SSL Operation with AppBeat DC

The process is as follows:

1. A secure connection is set up between the client and AppBeat DC, who acts as the server.
2. The client sends a request through the newly established secure SSL connection.
3. AppBeat DC sends the request to the server and receives the response from it over one of the existing optimized back-end TCP connections.
4. AppBeat DC encrypts the server's response and sends it to the client.
5. Requests and responses over new or existing non-secure connections continue to be handled and accelerated by AppBeat DC.

By fully offloading SSL tasks from the server, AppBeat DC saves the server significant processing resources, allowing it to continue to host the application optimally and with minimal overhead.

Integrated Functionality

AppBeat DC's SSL functionality is fully integrated with all other offloading services provided by the solution. When handling SSL connections, it processes the secure requests as if they were any other request processed by the device. The requests are sent to the servers over the optimized back-end connections and enjoy the same benefits as non-secure requests, such as connection consolidation and request/response buffering. Likewise, all secure content can also be processed by AppBeat DC's compression services. This means that SSL offload and compression can work together to reduce the amount of bandwidth consumed by each secure connection, the same benefits that clear connections utilize. Additionally, because all services are handled in dedicated, task-specific hardware modules, multiple services can be enabled at once with no effect on AppBeat DC's performance or functionality.

As more functions become available to AppBeat DC, this level of integration will continue, allowing all services to be used concurrently in a fully integrated manner. At the same time, the powerful underlying architecture of the solution will allow all functions to operate together without any degradation to the performance of the device.

Advanced SSL Functionality

AppBeat DC is also capable of providing more advanced SSL services that expand its applicability in secure environments.

- **Server-side SSL** allows AppBeat DC to maintain end-to-end security by transmitting secure client requests to the servers over back-end SSL connections. Although the server is once again dealing directly with secure connections, the overhead is significantly smaller due to the way AppBeat DC communicates securely with the server. First, a very small number of long lasting secure connections is established with each server, which means that the server doesn't have to deal with massive amounts of SSL handshakes. Second, the encryption keys used between AppBeat DC and the server can be weaker than those used with Internet clients, since this traffic is isolated within the network itself. This further prevents the back-end secure communications to place a significant burden on the server. With Server-side SSL, end-to-end security is preserved while the server is still significantly offloaded from the overhead of dealing with SSL connections.
- **Client Authentication** is an optional SSL process where a server can verify a client's identity, much like a client does to a server during the normal SSL handshake phase. The client can optionally present the server with a client certificate to confirm its credentials; a process that begins with the server requesting the certificate. The server validates the client certificate against a database of acceptable credentials and will only allow an authorized client to continue with its session. AppBeat DC includes this functionality in its SSL offload module, providing client authentication for the applications it is optimizing. This way, the full breadth of SSL features is available for the application, while the server continues to be relieved from dealing with these task-intensive functions.

Conclusion

AppBeat DC provides powerful SSL offloading and acceleration functionality that relieves the server from having to deal with the significant overhead that comes with handling secure communications. SSL functionality is fully integrated with all other server optimization features provided by AppBeat DC to the application, allowing multiple levels of offload to be performed on request/response chains, as necessary. At the same time, because of its unique and powerful task-specific, hardware-based architecture, all services can operate concurrently without any degradation in device performance.

CRESCENDO NETWORKS. 1.866.830.0400 • +972.3.538.5100 International • www.crescendonetWORKS.com

© 2008 Crescendo Networks. All rights reserved worldwide. No part of this document may be reproduced by any means nor translated to any electronic medium without the written consent of Crescendo Networks. Specifications are subject to change without notice. Information contained in this document is believed to be accurate and reliable; however, Crescendo Networks assumes no responsibility for its use,