



High Availability and Scalability of the Domain Boundary Controller

Both Xtradyne's Domain Boundary Controller (DBC) products (WS-DBC and I-DBC) provide linear scalability and unlimited support for various high-availability scenarios. A suitable solution guaranteeing both for any DBC installation can be chosen from a set of commonly accepted and widely tested approaches and technologies.

Xtradyne White Paper

Copyright © 2005-2010 PrismTech, All Rights Reserved.

Xtradyne is a registered trademark of PrismTech.

All other brand or product names are trademarks or registered trademarks of their respective owners.

Contents

1	Concepts and Terminology	3
1.1	Different Flavours of HA and Scalability	3
	High Availability and Scalability on the System Level	3
	High Availability and Scalability on the Application Level	3
1	High Availability and Scalability with a Traffic Redirector.	3
2.1	High Availability and Scalability as provided by the DBC.	4
2.2	Traffic Redirection: NAT versus Direct Routing.	5
	Network Address Translation (NAT).	5
	Direct Routing (DR)	6
3	High Availability Provided by Hot Standby.	7
4	Example: Deployment Considerations.	7
	Calculate Application Throughput	7
4.1	I-DBC Deployment Calculation Example	8
4.2	Deployment Requirements in Terms of Network Layout	8
	Replication and Stateful Failover.	8

Executive Summary

Full support for high-availability and scalability is an absolute necessity in most application scenarios of Xtradyne products. The Domain Boundary Controller (DBC) software product is vertically and horizontally scalable with nearly linear characteristics. The product is in use in several industries with extremely high requirements concerning high-availability, such as telecommunications and air travel. This document gives a technical overview on various approaches with various technologies how high-availability and scalability for the DBC can be achieved. The paper elaborates on what is actually meant by high-availability and scalability, how both are implemented in the DBC, how deployment should be planned, and what needs to be taken into account for configuring a highly available and scalable system.

1 Concepts and Terminology

The following sections will discuss the different mechanisms that the DBC architecture offers to achieve high availability and scalability. The ways in which high availability and scalability are tackled are closely related, therefore they are presented together. Let's first define what we mean by High Availability and Scalability in this context:

High availability (HA): The service of the DBC will still be provided even if a hard- or software component fails. This is achieved by replicating components of the DBC Proxy to eliminate single points of failure and providing health monitoring facilities. In case a component fails, a failover mechanism will use a replica of the failed component.

Scalability: Adapt the service of the DBC Proxy to fit higher requirements in terms of number of clients, throughput, or latency. Scalability can be achieved in several ways. The type of scalability presented here is obtained by operating multiple DBC Proxies in a cluster. A traffic redirector is used to distribute requests amongst DBC Proxies so that the load is shared.

1.1 Different Flavours of HA and Scalability

High availability and scalability can be provided in two ways. Either, the application takes care of failover and load-distribution itself (*application level*), or it relies on some external mechanism to provide the failover service (*system level*).

1.1.1 High Availability and Scalability on the System Level

To provide high availability and scalability on system level an external mechanism (on protocol level) is required. Such a mechanism is usually called *cluster management software*. A central part of this cluster management software the *traffic redirector*. A traffic redirector is a software add-on or dedicated device that employs a load-balancing algorithm to distribute client connections to a "cluster" of servers. Typically, this software presents the cluster hosts as a single virtual host and provides a single *virtual IP-address (VIP)* to the client. The traffic redirector of the cluster management software simply redirects network traffic from a failed or overloaded component to another working and less busy one, possibly in a way that is transparent to the client. The load of message processing is reduced on the individual DBC Proxy machines in the cluster, allowing the deployment of less expensive hardware. Examples for cluster management software are Sun Cluster 3 or Linux Virtual Server. Examples for traffic redirectors are Cisco CSS (Content Service Switch) or Cisco SLB (Server Load-Balancer) which is a feature of Cisco's IOS software and can be run on Cisco's switches.

1.1.2 High Availability and Scalability on the Application Level

The other possibility is to make the client aware of redundant components, thus providing high availability and scalability on the application level. This usually requires a higher development effort, but there are benefits: the application can be tailored more precisely to the requirements it has to fulfill. This includes but is not restricted to: faster failover, behavior based on knowledge about the failure state of components, better dynamic load balancing, improved stickiness of sessions. Besides, it saves the money for the cluster management software or traffic redirector.

2 High Availability and Scalability with a Traffic Redirector

To provide high availability and scalability several DBC Proxies can be operated in a cluster. Each DBC Proxy in a cluster shares its properties with any other DBC Proxy in the same cluster. In the standard

case (as depicted in figure 1), a traffic redirector will distribute the traffic from the clients amongst the DBC Proxies in this cluster. A typical cluster would consist of at least two DBC Proxies.

The clients reach the DBC service via the virtual IP address (VIP), i.e., the address of the traffic redirector (or load-balancer). The traffic redirector receives all the traffic and distributes the IP packets among the active DBC Proxies, based on the result of regular monitoring checks. If a DBC Proxy is overloaded or fails, the traffic redirector removes this DBC Proxy from its distribution list and forwards packets only to the remaining set of active DBC Proxies. The traffic redirector takes care that IP packets belonging to a single TCP connection are always directed to the same DBC Proxy. If a DBC Proxy fails in such a scenario, high availability is provided by terminating all TCP connections associated with the failed DBC Proxy and rerouting all new TCP connections to another DBC Proxy. The clients will see that their connections to the cluster are broken and will establish new TCP connections.

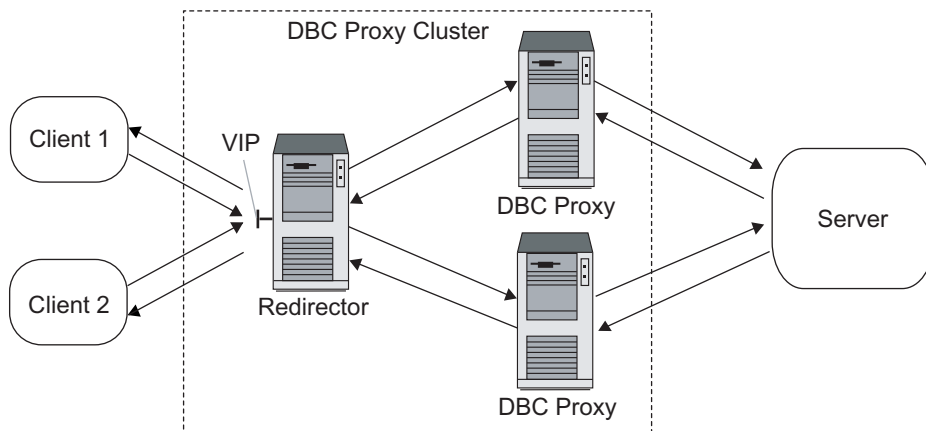


Figure 1 Multiple DBC Proxies with traffic redirector

2.1 High Availability and Scalability as provided by the DBC

The DBC is a distributed application. The DBC Proxies rely on the service provided by the Security Policy Server. If the Security Policy Server fails, the DBC Proxies can not longer provide their service to the clients. Therefore, to provide high availability a DBC installation can consist of multiple Security Policy Servers which constitute the Security Policy Server cluster. All Security Policy Servers in the cluster are configured identically so that any Security Policy Server can serve requests from any client. A typical deployment includes at least two Security Policy Servers.

In contrast to the typical client application, the DBC Proxy is aware of the fact that there are multiple Security Policy Servers available. If a Security Policy Server becomes unavailable, the DBC Proxy can automatically switch over to another one, without the need for help by a monitor agent. This simplifies the DBC configuration and saves additional load-balancers or monitoring components. In other words, the DBC Proxy uses system level HA and Scalability towards its clients, but application level HA and Scalability internally (as depicted in figure 2).

The Security Policy Server is also aware of multiple DBC Proxies in multiple clusters, also doing the failover automatically without the need for any external monitor agent.

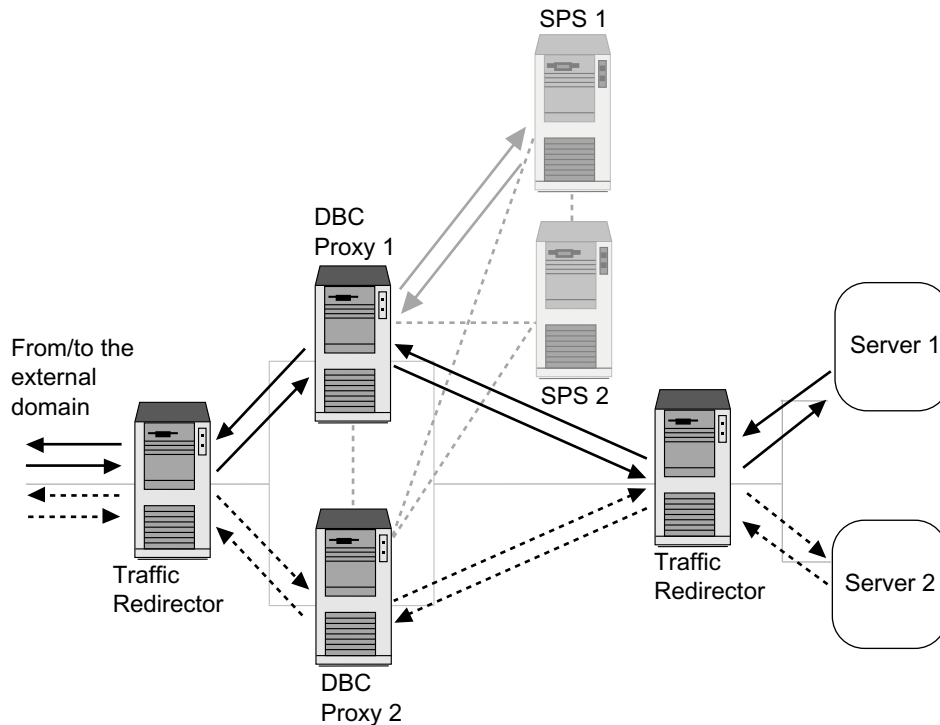


Figure 2 High Availability / Scalability as provided by the DBC

2.2 Traffic Redirection: NAT versus Direct Routing

There are various techniques for redirecting network traffic. The DBC can be used with *Network Address Translation (NAT)* and *Direct Routing (DR)*, as explained in the following sections. In any case, the DBC software configurations on all cluster machines must be identical, except, of course, for the local network addresses.

2.2.1 Network Address Translation (NAT)

The first redirection technique is Network Address Translation (NAT). The redirector is effectively a NAT router, providing a virtual address (VIP) for the DBC service of the cluster, as shown in figure 3, "Traffic redirection using a NAT router". A client packet targeted at this virtual address is routed to one of the DBC Proxies for processing, with the target address translated to the DBC Proxy's physical network address, the *Real IP address (RIP)*. Replies from the DBC Proxy are routed back to the redirector, which translates the physical originator address back to the virtual DBC Proxy address before routing the reply to the client.

The individual DBC Proxy machines in the cluster must use the redirector as the default gateway for reply routing. The DBC must be configured to use the virtual DBC Proxy address for proxification (assuming distribution is done for incoming client traffic). Apart from that, the configuration is the same as for a

single DBC Proxy solution. An advantage of this redirection technique is that the DBC Proxies do not have to be located in the same physical network or on the same VLAN.

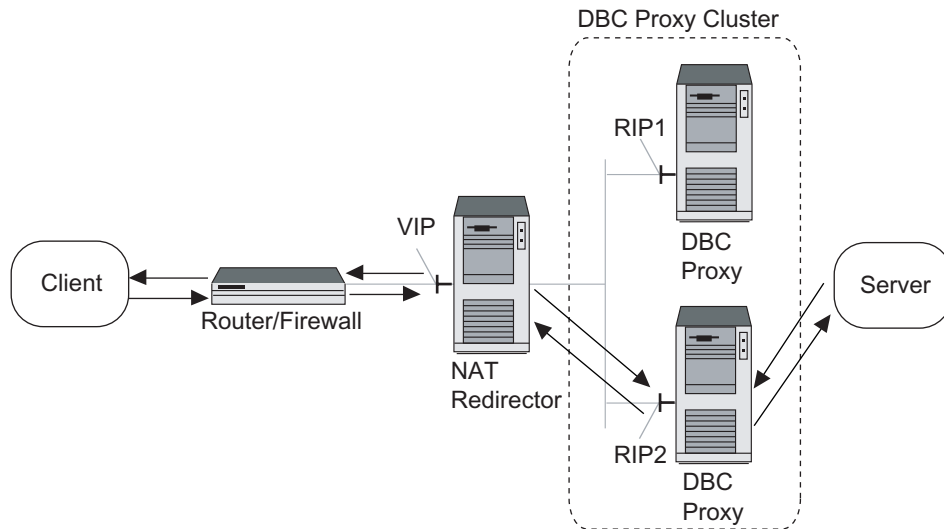


Figure 3 Traffic redirection using a NAT router

2.2.2 Direct Routing (DR)

The second redirection technique is Direct Routing (DR). Incoming and outgoing packets are routed on different paths (see figure 4, “Traffic redirection using direct routing”). All of the DBC Proxies in the cluster have the virtual address (VIP) configured as an alias address, typically on a loopback interface. The redirector forwards incoming client packets to one of the DBC Proxies for processing. Reply packets are routed directly to the client, bypassing the redirector.

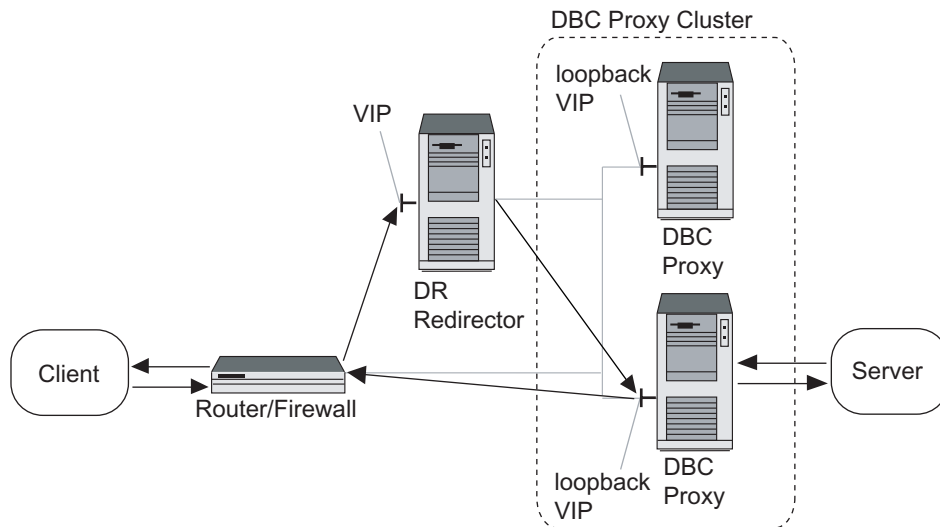


Figure 4 Traffic redirection using direct routing

This approach requires more complex configuration of the components. The router between the clients and the redirector must be configured to route all inbound client traffic to the redirector, but directly route outbound traffic. The individual DBC Proxy machines in the cluster must be capable of providing alias addresses on their loopback interface for configuring the virtual DBC Proxy address. The default gateway must be the router towards the client network. The DBC software must be configured to use

the virtual DBC address (VIP) as its external interface (assuming distribution is done for incoming client traffic). Apart from that, the configuration is the same as for a single DBC Proxy solution.

An advantage of this redirection technique is that it is faster than the NAT setup because replies are not routed via the traffic redirector. Note that outgoing connections must not come from the VIP address. A disadvantage of the Direct Routing setup is that the DBC Proxies have to be located in the same physical network.

3 High Availability Provided by Hot Standby

The hot standby approach is based on a single machine hosting the *primary DBC installation*, which serves requests on a virtual IP address and performs normal message processing. In case of failures, a secondary (or “standby”) DBC machine takes over and guarantees uninterrupted service to clients.

This approach relies on a secondary DBC host *monitoring* the primary DBC host, and on network-level functionality to take over the virtual IP address used by the primary host. This functionality is offered by a separate failover package, which is included in the DBC distribution and combines DBC-specific monitoring and failover functions.

4 Example: Deployment Considerations

The optimal deployment of a load-balanced DBC architecture depends on several interdependent variables that must be considered during the planning phase. The following procedure may help to determine the requirements for a given scenario in terms of number of machines in the cluster, network bandwidth, and traffic redirector capabilities.

If availability is an issue, remember that *all* involved components (server, redirector, networks) must be laid out in a redundant form.

4.0.1 Calculate Application Throughput

5. Measure the average throughput required by a typical interaction in your application, between a single client and the server, without the DBC Proxy.
6. Calculate the total required throughput from the anticipated number of concurrent sessions and the measured single-session value.
7. Make sure the network deployed between the client and the server is capable of handling the total throughput. If not, you will have to upgrade the network first.
8. Make sure your server is capable of handling this total throughput. If not you will have to find a load balancing solution for this problem first.

4.0.2 Calculate DBC Requirements

1. Select a traffic redirector product that is capable of handling the total application throughput. Make sure it has at least minimal monitoring capabilities, if not add a compatible monitor product.
2. Select a hardware platform for the DBC cluster machines.
3. Measure the maximum throughput that a DBC on the selected platform can provide. You can do this by running an increasing number of concurrent sessions of your application through the DBC Proxy, and finding the strongest downward bend in the resulting performance graph.
4. Calculate the total number of DBC Proxies required from the total throughput and the maximum throughput of the single DBC Proxy. You may need to repeat these last three steps to optimize the cost-to-performance balance.

After finishing the process, you may want to estimate the performance of the scenario under peak load. For that purpose, repeat the application throughput calculation, but this time measuring the *maximum* single session throughput. Compare the resulting throughput against the capacities of your server, network, and redirector. Divide the throughput by the number of planned DBC machines, and check their performance graphs with this load.

4.1 I-DBC Deployment Calculation Example

This section applies to the **I-DBC** only.

Let's consider an application that requires an average throughput of 200kBit/sec and a maximum throughput of 300kBit/sec in each direction for a typical IIOP session, i.e., between a single client and the CORBA server. The session consists of 6 request/reply round trips each of which contains 4 kByte (ca. 192kBit/sec in each direction).

We anticipate a requirement for 250 parallel sessions, so the required average throughput in each direction is 48MBit/sec with a peak of 75 MBit/sec and 1500 messages per second. The total required average throughput (in both directions) is 96MBit/sec with a peak of 150MBit/sec and 3000 messages per second.

We have an existing infrastructure built on 100MBit/sec FastEthernet, which will be capable of sustaining this load both in the average and maximum case, and thus is sufficient for this application. The CORBA server runs on high-end hardware and is also capable of handling this load.

For this example, we assume the traffic redirector deployed can handle the 100MBit/sec full duplex of the network without measurable performance impact.

Performance tests were done with standard PC hardware for the I-DBC platform (because it has a good value to performance ratio). Specifically, a Dual-Pentium III 866Mhz system with 512MB RAM and two quality network interface cards. This machine was capable of handling a peak throughput of 85MBit/sec at 4 kByte per IIOP message with 2700 IIOP messages per second.

Consequently, we need two of these machines to handle the average application throughput. Assuming a good load balancing algorithm, each machine would handle 48MBit/sec in this case, running at 64% load, and handling 1500 IIOP messages per second. The two machines will also be capable of handling the maximum application throughput. Each would handle 75MBit/sec, running at 88% load.

4.2 Deployment Requirements in Terms of Network Layout

When planning the system, take the following requirements concerning the network into account: The Security Policy Servers must be able to contact each other directly. This is necessary for the synchronization of configuration data and state information between the Security Policy Servers.

Each Security Policy Server in the cluster must be able to contact any I-DBC Proxy directly. This means that the connection is made directly to the respective I-DBC Proxy and that no redirector must be interfering with the connection. This is absolutely necessary to make sure every I-DBC Proxy will be configured in the startup process.

4.2.1 Replication and Stateful Failover

The I-DBC offers a feature called "Replication" which enables stateful failover. Stateful failover means that any hardware or software failures will go completely unnoticed to the client. Replication enables multiple I-DBC Proxies to share their state. Newly proxified IORs will be multicast to the other I-DBC Proxies. This enables stateful failover as any I-DBC Proxy will have all the necessary information available to serve any client request.

If replication is to be done a multicast address must be enabled on every I-DBC Proxy in a cluster.

About Xtradyne Products

Xtradyne is PrismTech's security software division offering application-level proxies and security gateways built specifically to address the needs of large corporations and enterprises. Xtradyne's runtime products help companies extend their automated business processes beyond corporate firewalls towards partners without compromising security. Application-level security implemented by Xtradyne's turnkey security solutions provide a very economic way to address the increasing needs for security in J2EE multi-tier architectures and Web services. Due to Xtradyne's unique approach, additional security can be added with minimal time to market, extremely reduced reengineering costs, and very low total costs of ownership.

About PrismTech

Founded in 1992, with offices in the USA and Europe, PrismTech is a privately held software products company. PrismTech serves international Fortune 500 customers in the telecommunications, data communications, defense and aerospace sectors. PrismTech is an acknowledged leader in distributed and wireless software infrastructure, with solutions ranging from wide-scale integration to embedded real-time systems, supporting applications from operations support systems (OSS) through to software defined radio (SDR). For additional information about PrismTech, visit the web site at www.prismttech.com.

Further information:

USA Corporate Headquarters PrismTech Corporation

400 TradeCenter, Suite 5900
Woburn, MA, 01801
USA
Tel: (1) 781-569-5819

European Headquarters PrismTech Limited

5th Avenue Business Park, Team Valley
Gateshead, Tyne & Wear, NE11 0NG
United Kingdom
Tel: +44 (0) 191-4979900

For product or technical questions, please contact our Customer Response Center,
Email: [**crc@prismtech.com**](mailto:crc@prismtech.com)

For licensing and pricing questions, please email to [**sales@prismtech.com**](mailto:sales@prismtech.com)