MIDDLEWARE PROXY: A REQUEST-DRIVEN MESSAGING BROKER FOR HIGH VOLUME DATA DISTRIBUTION

W. Sliwinski, I. Yastrebov, A. Dworak, CERN, Geneva, Switzerland

Abstract

Nowadays, all major infrastructures and data centres (commercial and scientific) make an extensive use of the publish-subscribe messaging paradigm, which helps to decouple the message sender (publisher) from the message receiver (consumer). This paradigm is also heavily used in the CERN Accelerator Control system, in Proxy broker - critical part of the Controls Middleware (CMW) project. Proxy provides the aforementioned publish-subscribe facility and also supports execution of synchronous read and write operations. Moreover, it enables service scalability and dramatically reduces the network resources and overhead (CPU and memory) on publisher machine, required to serve all subscriptions. Proxy was developed in modern C++, using state of the art programming techniques (e.g. Boost) and following recommended software patterns for achieving low-latency and high concurrency. The outstanding performance of the Proxy infrastructure was confirmed during the last 3 years by delivering the high volume of LHC equipment data to many critical systems. This work describes in detail the Proxy architecture together with the lessons learnt from operation and the plans for the future evolution.

INTRODUCTION

Already in 2008, at the time when LHC accelerator started the beam operation, it became evident that the existing controls infrastructure was not capable to serve the continuously increasing demands of many data intensive applications. The front-end computers (FECs), running the real-time LynxOS system, with 1 CPU and limited memory, responsible for the control of equipment and data acquisition, were the main source of a major performance and scalability bottleneck. Therefore, the Middleware team came with the proposal of introduction of a Proxy server, which would decouple handling of subscriptions to all interested users, decrease use of resources on FECs and allow for better scalability of data distribution channels. The Proxy infrastructure was developed in close collaboration with equipment groups and operation team and it was deployed for all major equipment systems in LHC and partially for other CERN accelerators.

REQUIREMENTS

The following technical requirements were defined for the middleware Proxy:

- Proxy (acting as middle-tier server in peer-to-peer environment) should be integrated transparently into the control system, so that client applications should not experience any difference when operating directly with remote device servers or via a Proxy. Moreover, it must use the RDA2 [1] framework for the client and server sides.
- As a middle-tier server, Proxy intercepts every request coming from a client to device server (Get, Set, Subscribe). This imposes very tight constraints in terms of performance: Proxy should not significantly impact performance of the overall communication.
- The main Proxy responsibility is to decouple publishers from subscribers for handling of subscriptions. Proxy should implement grouping of subscribers and broadcasting of the subscription updates.
- Proxy architecture must be scalable and highly concurrent to serve many independent clients simultaneously.
- Asynchronous processing of the subscription updates is required in order to guarantee non-blocking communication, especially when slow client consumers are present.
- Proxy should preserve ordering of the subscription updates for each subscriber.
- Proxy must be integrated with the existing Role-Based Access Control (RBAC) [2] mechanism for the CERN control system.
- Proxy software may run permanently for very long period without interruptions, therefore it is important to provide advanced diagnostics and monitoring capabilities in order to be able to inspect its state at any time.

PROXY ARCHITECTURE

The development of Proxy started in 2008, when the core part of the LHC control system was already in place. It was built on top of the existing components and became an integral part of the middleware infrastructure and the CMW project.

Client applications operate remote devices either directly (2-tier mode) or via middle-tier servers (3-tier mode). Moreover, there can be several middle-tier servers between a particular client and a device server.

Proxy supports two types of communication: request-reply (Get and Set calls) and subscriptions (Subscribe calls). Proxy acts as an intermediate component between a client application and a device server, which controls some physical equipment. Figure 1 shows the overall architecture of the control system including Proxy servers.

ISBN 978-3-95450-139-7

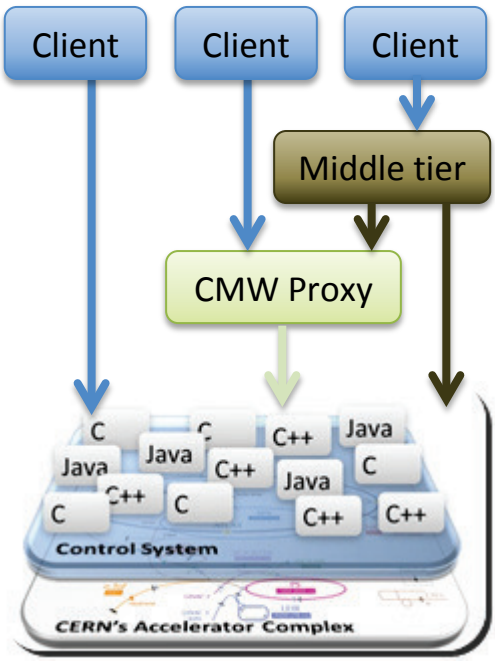

Figure 1. Proxy in the control system

Request-Reply Channel

For the request-reply calls, Proxy performs additionally several operations as described subsequently.

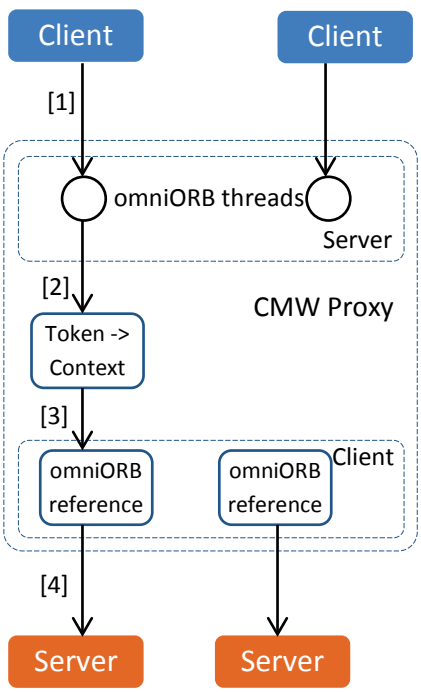

Figure 2. Proxy data flow for request-reply calls

When a client application sends a Get or Set request to a certain device (1) it is handled by the Proxy server-side. The server-side is based on the RDA2 library, which is implemented on top of the omniORB product [3], i.e. CORBA C++ ORB implementation. Currently, we use thread-per-connection mode, which allocates one thread per each physical client connection. A dedicated thread handles all requests from the given connection (2). Proxy inserts client token, used for device access authorization, in the context of each request (3), such that the original client token is propagated on "per-operation" basis. Next, the request is sent synchronously to the device server, where authorization and the final processing are performed (4). The detailed data flow for request-reply calls is depicted in Figure 2 above.

Subscription Channel

For the subscription calls, Proxy performs much more additional processing. When a client requests to establish a subscription (1), Proxy first performs grouping and matching of the incoming subscription with the already existing ones (2). This means that Proxy establishes at most one physical subscription to the device server for a given property and if there are several subscribers to the same property, Proxy broadcasts subscription updates to all of them when a new update arrives from the device server. Therefore, when a client subscribes to a property, for which there is already an established subscription, Proxy adds the new subscriber to the existing group and does not send the subscription request to the physical device. This however means that new subscribers are not guaranteed to receive so called "first-update" after the subscription grouping, because the device server is not aware of the new subscribers. In order to solve this issue and always provide the first-update (3), Proxy performs a Get call for the grouped subscription before further processing of the request (4). After receiving the first-update, Proxy sends it synchronously to the subscriber and adds it to a particular group. Obviously for the first subscription in the group handling is different: the first-update step is not needed, since device server can provide the first-update on its own. Instead, Proxy establishes physical subscription to the device server using Proxy authentication token (5). Server authorizes the subscription request (6) using the provided Proxy token, which should contain the CMW-PROXY role. As a consequence, RBAC access rules for the servers behind a Proxy must explicitly allow the CMW-PROXY role to establish subscriptions. After a subscription is confirmed, server periodically publishes updates to the listeners (7). On the Proxy side, there is a dedicated dispatcher per each client connection that contains message queue and a dedicated processing thread. Each subscription update is pushed into corresponding message queue (8) and processed (delivered to the client) by dedicated dispatcher thread (9). This architecture allows us to:

- Solve the "slow client" problem. If there are slow consumers in the system, they do not block other clients, because of a separate message queue for each client;
- Preserve ordering of the subscription updates;
- Apply notifications drop policy for slow consumers only;

ISBN 978-3-95450-139-7

- Have detailed statistics and diagnostics per each client and each subscription.

Figure 3 depicts the data flow for subscription calls.

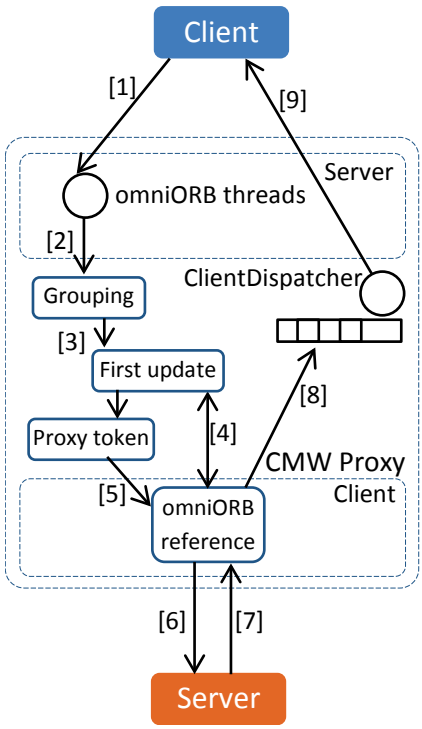

Figure 3. Proxy data flow for subscriptions

Configuration and Deployment

An important aspect for operational use of a service is configuration support and ease of deployment.

Any device server, implemented using the RDA2 framework, can be accessed via a particular Proxy, thanks to several configuration methods, which can be applied, one at a time:

- Mapping through the Controls Configuration Database [4] – this is the default approach, which requires a given device server to be mapped to a particular Proxy in the database. One Proxy can be mapped to many device servers, but a particular device server can be mapped to at most one Proxy. This approach provides full transparency for all involved parties, i.e. client side, Proxy and device server. The client calls are redirected via a Proxy thanks to the CMW Directory service resolution, which is based upon the latest database mappings. This is also the preferred way for running all operational Proxy instances.
- Programmatic mapping using RDA2 API – this approach can be used to override the database mapping, when an explicit Proxy server name is known in advance (e.g. command line argument or configuration file). It is used in the continuous integration environment, where different types of redirections are tested.
- Mapping through the system properties – this approach is most often used in test setup, when both previously described configurations (database driven and programmatic) have to be overridden temporarily for a test purpose. It can be also used during the integration testing of several controls components.

Having in place several ways to configure device server to Proxy mapping, allows for a non-intrusive deployment of new Proxy servers even during the beam operation.

Access Control

Proxy servers act as an intermediary layer between clients and actual device servers. The major security issue in this model is how to enforce the access control for subscriptions.

In order to address this problem authentication for Proxies was introduced. The purpose of authentication is to verify the digital identity of a principal. If the authentication process succeeds, its result is a digitally signed authentication token that is returned to the application. The token is a short-term uniform substitute of the real credentials. It is issued by the central authentication server, which can reliably verify the user's identity [5]. At start-up, each Proxy performs authentication by location, without using explicit credentials and obtains a token that contains the CMW-PROXY role. RBAC access rules for devices working behind a Proxy must allow the client subscriptions on desired properties for that role. This approach has several advantages. First, being very simple, efficient and non-intrusive, it enforces access control in a single place. Second, it helps equipment specialists to impose usage of a Proxy for certain device servers thus preventing direct client access and limiting the performance problems [2].

Diagnostics and Monitoring

For diagnostics and monitoring of any CMW server a dedicated GUI application was developed, called CMW Admin, that can query any server for information about its state. Proxy is a CMW server inherently and it also exposes common properties, e.g.: configuration, state, logging, connected clients, etc. Additionally, it exposes specific administrative properties to monitor:

- State of each grouped subscription - this property reports detailed information about grouped subscription, status of each subscriber, number of updates and timestamp of the last subscription update.
- State of each connected client together with state of the allocated thread and message queue for that client.

Figure 4 presents the detailed diagnostic view of the grouped subscriptions together with related information about the subscribed clients.

ISBN 978-3-95450-139-7

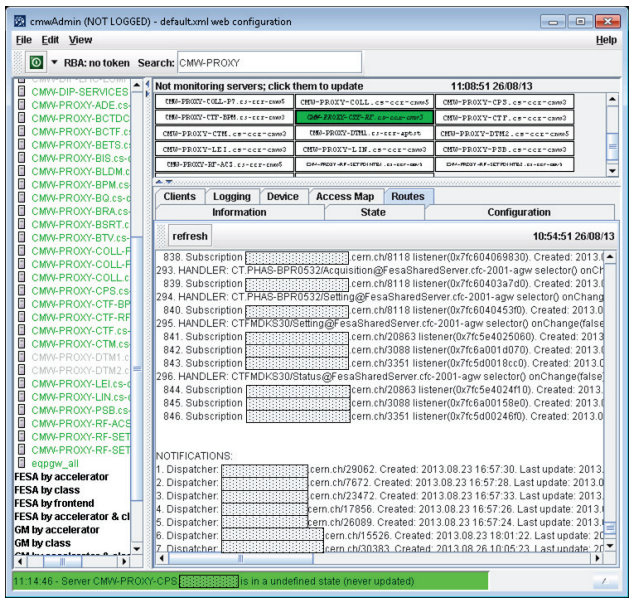

Figure 4. Proxy diagnostic view in CMW Admin GUI

STATUS OF THE PROJECT

Proxy was successfully deployed and commissioned in LHC operation in 2009. The system passed many centrally organized tests. The feasibility, performance and overhead of using Proxy were experimentally evaluated. The results show that the overhead is acceptable and the chosen approach can be effectively used in the CERN control system.

Currently there are 26 operational Proxy servers and additionally 2 instances running in the test environment, i.e. the Controls Testbed [5]. Test configuration allows us to perform integration testing without interaction with the production environment. The outstanding performance of the Proxy infrastructure was confirmed during the last 3 years by delivering high volume of LHC equipment data to many critical systems. As a result, even the constrained front-end computers, operating with limited resources, were able to deliver data to many critical applications, which was not possible before introduction of the Proxy.

Nevertheless, there are still few areas, where current implementation can be improved and extended in order to expand the area of its applicability.

Limitations

Proxy was built on top of the existing infrastructure: RDA2 and RBAC. Proxy development started when those components were already implemented and we could not change API and architecture of the mentioned libraries. As a consequence, there are several limitations with the current version of Proxy.

First important limitation is copy overhead. For each incoming request Proxy does deserialization from the network representation into Data object, which is then serialized again into the network form. This conversion is not necessary, since Proxy does not change the data (it may change only context) but it is expensive. Performance analysis showed that Proxy wastes around 50% of CPU time doing this data transformations.

Another limitation is the lack of support for permanent subscriptions done via Proxy with client authentication token. The workaround is to always establish subscription through Proxy with Proxy's authentication token. As a consequence access rules of the device servers behind the Proxy must be modified in order to support authorization of the middle-tier Proxy server.

FUTURE PLANS

The most important objective for Proxy in 2013/2014 is integration with the new RDA3 framework [7]. The new Proxy version will allow for communication between clients, both RDA2 (based on CORBA) and RDA3 (based on ZeroMQ), and RDA3 device servers. This would help to organise a smooth introduction of the new RDA3 device servers without changing immediately code of the client applications. The new RDA3 provides better integration with RBAC and abstractions for building middle-tier services like Proxies.

In the next major Proxy version we plan to optimize the performance, to make the product more scalable, reliable and responsive. Our goal is to remove unnecessary copy overhead by avoiding expensive serialization and deserialization thanks to the architecture of the new RDA3. We also plan to achieve better integration with the access control mechanism in order to:

- Get rid of "Proxy RBAC token" and perform all the communication using clients token;
- Eliminate the need to modify access rules on the device server side to allow going through a Proxy;
- Make communication fully transparent by using advanced session mechanism, so that server knows all of its clients, even if they are connected via a Proxy.

REFERENCES

[1] N. Trofimov *et al.*, "Remote Device Access in the new CERN Accelerator Controls middleware", ICALEPCS'01, San Jose, California, USA.

[2] I. Yastrebov *et al.*, "Status of the RBAC infrastructure and lessons learnt from its deployment in LHC", ICALEPCS'11, Grenoble, France.

[3] omniORB: http://omniorb.sourceforge.net

[4] Z. Zaharieva *et al.*, "Database Foundation for the Configuration Management of the CERN Accelerator Controls System", ICALEPCS'11, Grenoble, France.

[5] A. Petrov *et al.*, "User Authentication for Role-Based Access Control", ICALEPCS'07, Knoxville, Tennessee, USA.

[6] J. Nguyen Xuan *et al.*, "Testbed for Validating the LHC Controls System Core Before Deployment", ICALEPCS'11, Grenoble, France.

[7] A. Dworak *et al*, "Middleware trends and market leaders 2011", ICALEPCS'11, Grenoble, France.

ISBN 978-3-95450-139-7