Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems.
March 22-24, 2011, Washington, DC, USA.

An extensible DDS-based monitoring and intrusion detection system

Fernando Garcia-Aranda[a,*], Javier Sanchez-Monedero[b] and Juan M. Lopez-Soler[a]

[a]Department of Signal Theory, Telematics and Communications, University of Granada (Spain)

[b]Department of Computer Science and Numerical Analysis, University of Cordoba (Spain)

[*]**email:** fgaranda@ugr.es

Outline

Outline

Introduction

Systems built around a centralized computing model are being superseded by new systems that are suitable for deployment in more **distributed** computer environments.
This is driving an increase in the demand for technology to supervise application, system and network health.

Although multiple solutions already exist, they tend to...

- be monolithic (e.g., specifying custom data-models and protocols to distribute the information).
- serve a single purpose (e.g., performance monitoring vs. IDS).
- be limited in scalability (e.g., based on a centralized client-server model).

Introduction | Our solution (I)

We propose **Cave Canem**: an open distributed framework for the collection and integration of global system status, events, and alerts.

Motivating scenario (I)

The Spanish National Lab PASITO is a common network infrastructure that performs multidisciplinary research on future networks and massive computing issues.

$\Rightarrow$ This environment requires an **efficient and global distributed monitoring** framework to provide a complete picture of the health status of the overall system.

$\Rightarrow$ Different sites may be interested in different monitoring issues, and these interests may arise at different locations with **diverse requirements**.

Introduction | Our solution (II)

Motivating scenario (II)

The following requirements must be satisfied:

Efficient data filtering e.g., a system administrator at a given site is interested in retrieving all alerts from a given set of sensors, whereas another system administrator is only interested in messages exceeding a given threshold.

Space-time decoupling Information producers and subscribers must be decoupled in space and time.

The communication model must fit the nature of the problem Monitoring large number of distributed sensors requires the retrieval of information about events as soon as they occur.

Information priority Information produced by sensors can have different importance levels.

Introduction | DDS (*Data Distribution Service*) (I)

To address these issues, Cave Canem framework exploits the advantages of the OMG Data Distribution Service (DDS).

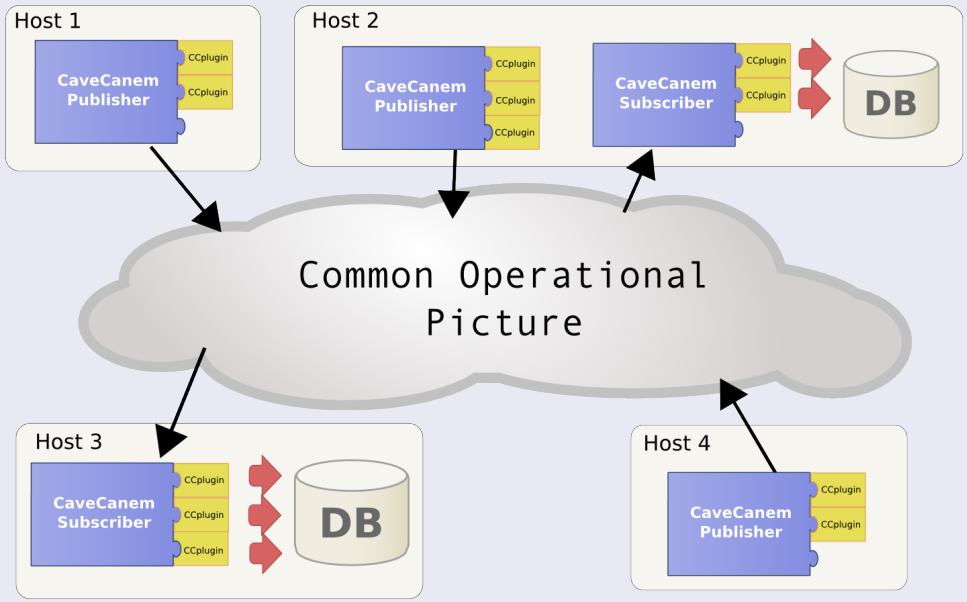
Advantages of the use of DDS (I)

- DDS provides access to the formation from anywhere in the system using a data-centric publish-subscribe (DCPS) model.
- DDS implementations can operate completely in the P2P mode, avoiding the need to deploy any central services.
- The DCPS model allows the middleware infrastructure to reflect the essential aspects of the information model, cache the required information, provide time and content-based filtering, and ultimately deliver the information to the applications with the correct QoS.

Introduction | DDS (*Data Distribution Service*) (II)

Advantages of the use of DDS (II)

- The DDS data-centric middleware constructs a Global Data Space (GDS) in which the Cave Canem can create a Common Operational Picture (COP) that can combine information from disparate sensor classes. This COP makes the information available for observation and analysis in a seamless and efficient manner.



Outline

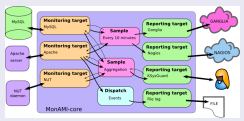
Related Work

Nagios Computer network monitoring system that monitors the status of hosts and services, and generates alerts upon the occurrence of a relevant change.

Pandora FMS Application for the monitoring of systems, devices, applications and services.

AlienVault OSSIM is a set of tools designed to aid network administrators in dealing with computer security, intrusion detection and OS monitoring issues.

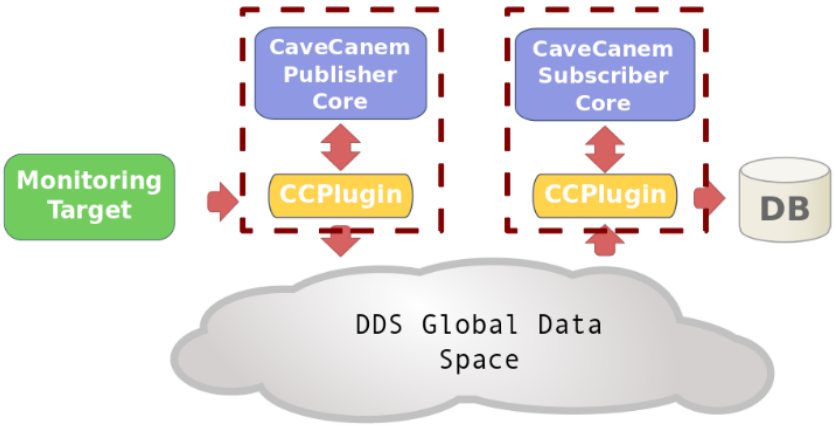
MonAMI is a framework for the collection and dissemination of monitoring information.



Outline

Architecture | Global overview

CCPublisher provides a universal monitoring agent. It collects and disseminates monitoring information.

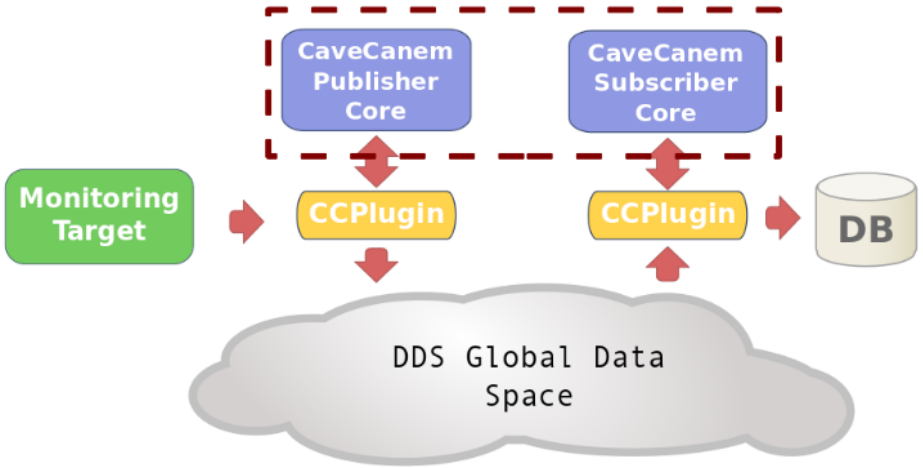


CCSubscriber stores information in any type of database management system or log file for additional online or offline processing.

Advantages of the architecture

- It decouples producers and consumers of information in space and time ⇒ Scalability.
- It makes easy to implement redundant information processing and visualization points.

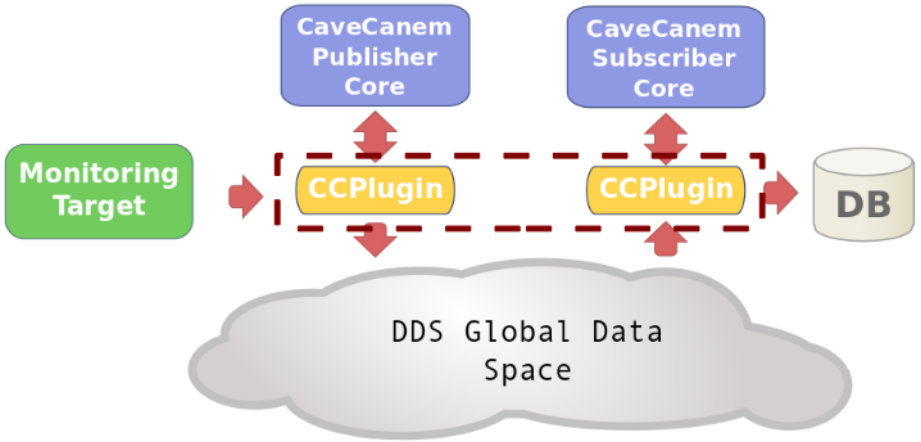
Architecture | CCPublisher and CCSubscriber core apps.



Tasks

- To deal with minor configuration issues, using a set of XML files. These files include identifications of the plugins to be loaded by CCPublishers and CCSubscribers.
- To provide a base API for the development of plugins. This API allows the system to load plugins and to access methods of DDS publication and subscription.

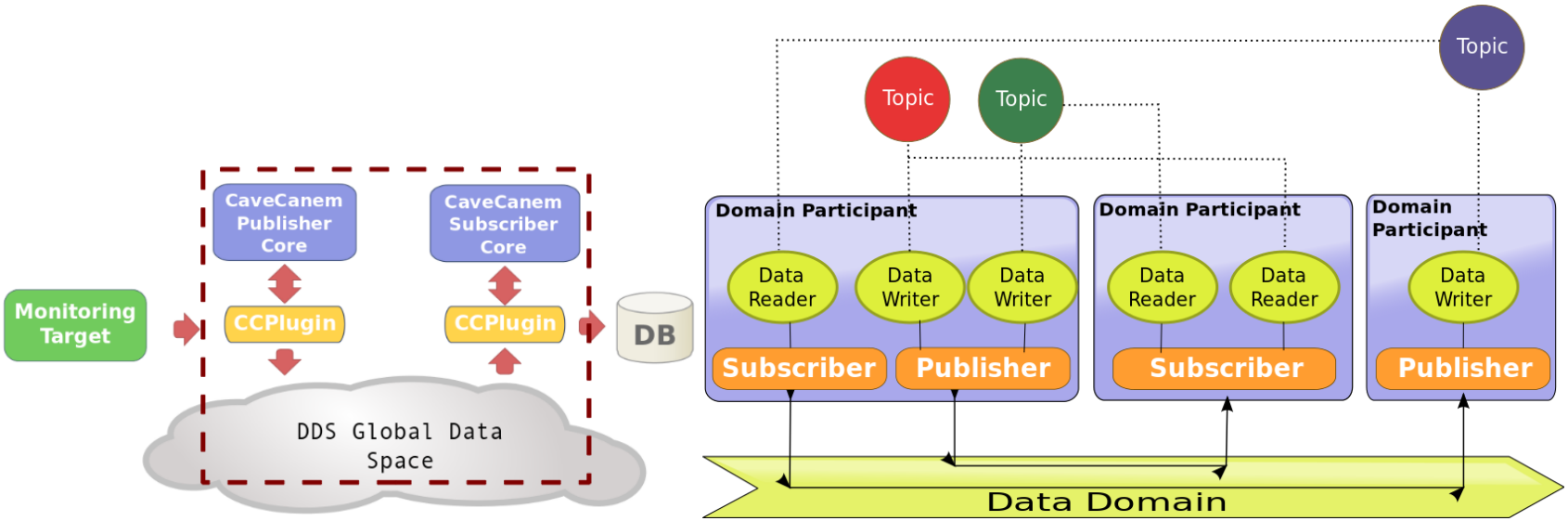
Architecture | CCPublisher and CCSubscriber plugins (I)



Tasks

- Each CCPublisher plugin represents a single monitoring agent as it collects data from a given monitoring target. The collected information is published in the DDS GDS using its normalized template (DDS Topic).
- Each CCSubscriber plugin collects information from the DDS GDS related to a given DDS Topic and stores it in any type of database or log file.

Architecture | CCPublisher and CCSubscriber plugins (II)



DDS entities in Cave Canem

CCPublisher Plugins

- DDSDomainParticipant
- DDSPublisher
- DDSTopic
- DDSDataWriter

CCSubscriber Plugins

- DDSDomainParticipant
- DDSSubscriber
- DDSTopic
- DDSDataReaderListener

Outline

DDS features used (I)

This section discusses the utilization of DDS features in the Cave Canem framework to improve its functionality and performance.

(1) Information space and time decoupling

DDS provides a publish-subscribe information distribution model, which decouples publishers and subscribers in space, platform and time.

In time The selection of the adequate *Transient* or *Persistent* policies, allows the monitoring system to dynamically add new CCSubscribers that can access data produced in the past.

In space The CCSubscribers and the CCPublishers can be located at any node in the network, making it easy to implement redundant information processing and visualization points.

DDS features used (II)

(2) Content Filtered Topics

In addition to the regular Topics, the DDS also defines Content Filtered Topics that allow the definition of filter expressions for data samples.

$\Rightarrow$ Only samples that match the filter are received by the subscribers and forwarded to the application.

Definition of a filter expression in CCSubscriber

```
<my_disk>
  ...
  <qos_library>NddsQosProfilesLibrary</qos_library>
  <qos_profile>default</qos_profile>
  <filter_expresion> used_per > 70 </filter_expresion>
  ...
</my_disk>
```

DDS features used (III)

(3) QoS parameter selection

The following selected DDS QoS parameters are specifically relevant to improving the Cave Canem framework's performance:

Reliability It can be set up either as **Best Effort**, which suppresses the retransmissions of error data but reduces the end-to-end latency, or as **Reliable**, which ensures the error free and ordered delivery of data samples.

Time-Based Filter This parameter may be used in the reception of sensor values that are not needed instantly even if the *DataWriter* publishes data faster than the minimum separation period.

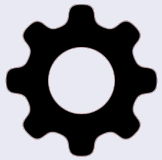
History It specifies the number of data samples to be stored for later delivery in DDS.

Outline

Implementation

Plugins developed

Configuration system



Storage system

Web UI

Implementation | Plugins developed (I)

System Monitor

This plugin handles basic information about the status of systems that contain the CCPublisher. It manages the CPU load, the temperature, and the memory and swap usage.

Disk

This plugin handles information related to the different file systems used in the monitoring target, e.g., its name, mount directory, and usage.

Implementation | Plugins developed (II)

Net Load

This plugin provides information related to network interfaces used by the monitoring target including their hardware and network address, and the data concerning their traffic.

Snort (IDS)

We define a common IDS alert interface for the interoperability of IDS-related plugins. To define this common interface, our prototype includes a plugin called Snort, which deals with information extracted from the IDS.

Implementation | Plugins developed (III)

Snort (IDS) (II)

This common IDS alert interface includes information related to the alert, its source and its target.

```
struct ids_alert {
  host_info_t host; //@key
  node_info_t source;
  node_info_t target;
  timestamp_t ts;
  string msg;
  string timestamp_string; //@key
};
```

Implementation | Configuration system (I)

CCPub. and CCSub. are configured using a set of XML files.

General configuration in CCSubscriber

```
<general>
  <!-- here in CCPublisher it is also configured the
       publishing period !-->
  <qos_file>cavecanem_publisher_qos.xml</qos_file>.
</general>
<database>
  <server>localhost</server>
  <name>cavecanem</name>
  <user>cavecanem</user>
  <password>cavecanem</password>
</database>
```

Implementation | Configuration system (II)

Plugins configuration in CCPublisher

```
<plugins>
  <my_disk>
    <plugin_id>my_disk</plugin_id>
    <plugin_type>disk</plugin_type>
    <lib_name>libcavecanem_publisher_disk.so</lib_name>
    <domain_id>3</domain_id>
    <topic_name>filesystem</topic_name>
    <qos_library></qos_library>
    <qos_profile>default</qos_profile>
  </my_disk>
    ...
</plugins>
```

Implementation | Configuration system (III)

QoS configuration

Besides the general and plugin configuration, our system supports XML-based QoS configuration. This functionality, provided by the DDS middleware, allows us to update the QoS configuration without rebuilding or redeploying the application.

Using a Time Based Filter

```
<example_plugin>
  ...
  <qos_library>experiments</qos_library>
  <qos_profile>tbf</qos_profile>
  ...
</example_plugin>
```

Implementación | Storage system

The design provides great flexibility in the development of the storage of information due to the absence of restrictions on the inner implementation of the reception methods.
⇒ We are going to describe a storage example using a MySQL database.

Storing the monitored data in a MySQL database (its tables)

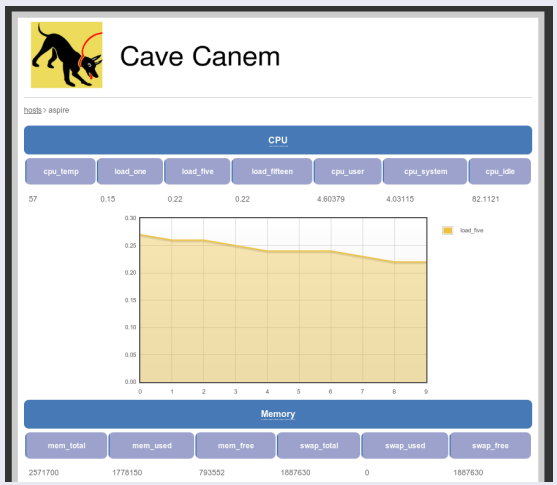
hosts	Stores information about the hosts that are publishing monitoring information.
system_mon.	Stores information related to this topic (plugin) [RR].
disk	Stores information related to this topic (plugin) [RR].
ids	Stores information related to this topic (plugin).
net_load	Stores information related to this topic (plugin) [RR].

Implementation | Web UI

To perform later online and offline processing of the information collected by CCSubscriber we have developed a web UI that gets information from its databases.

Main contributions

- It provides a centralized environment to analyze the collected information.
- It provides graphical information related to some parameters that allows the evaluation of the evolution of the monitored data.

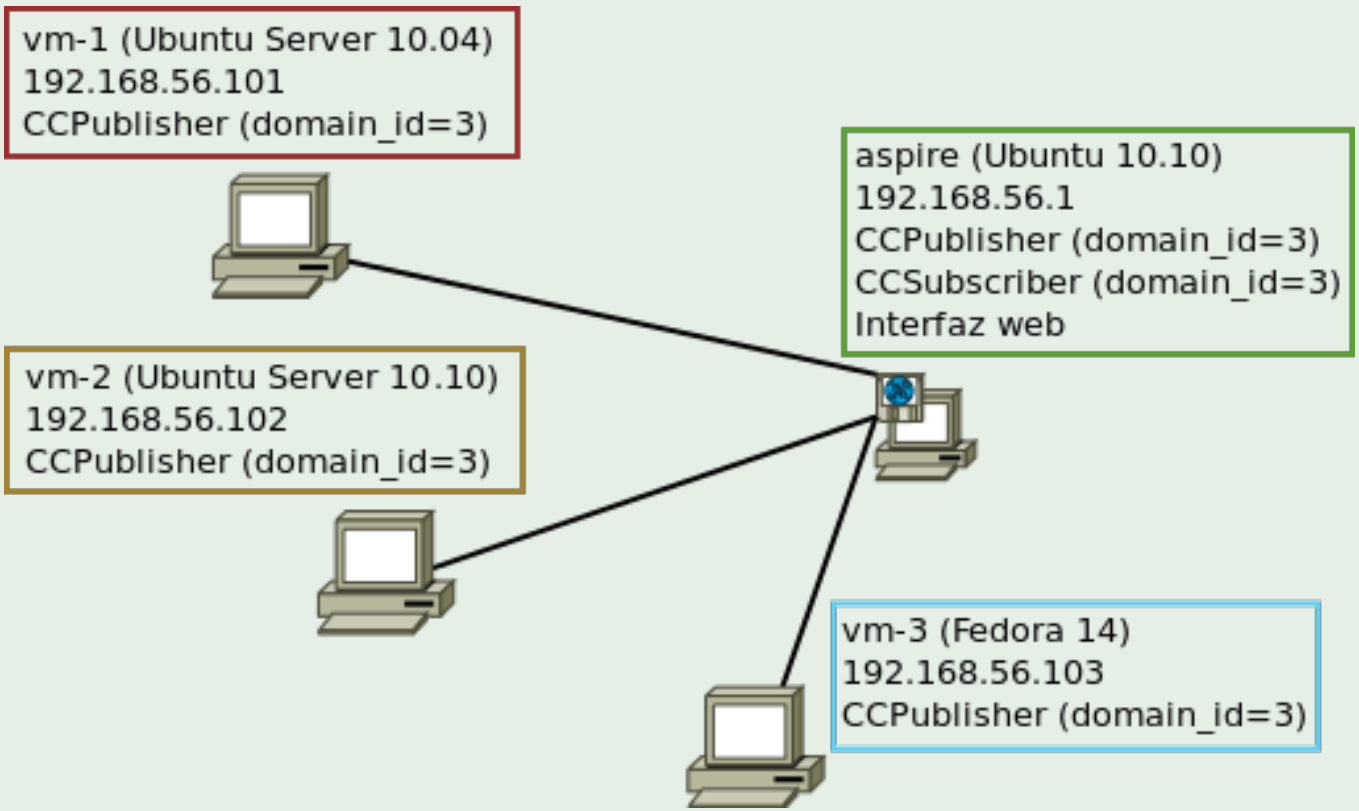


Outline

Demo

To demonstrate the behavior of Cave Canem, we are going to run it on a demo scenario.

Demo scenario



Outline

Conclusions (I)

- We have proposed the Cave Canem framework for monitoring system health.
- Our proposal separates the information model from the information distribution mechanisms and uses a pure P2P distributed approach that avoids the need to deploy a central service.
- Cave Canem framework adopts a DCPS communication model based on the OMG DDS standard.
- Cave Canem framework defines a COP model that collects and integrates the monitored data from different sensors, systems and networks to facilitate correct and timely management decisions.

Conclusions (II)

- The framework exploits several features of DDS and uses QoS policies such as reliability and filtering. Thus, the Cave Canem COP normalizes the data producers (sensors) and consumers in time, space and platform.
- To ensure extensibility, the design of the Cave Canem framework is based on a flexible plugin architecture.
- We have implemented a Cave Canem prototype that fulfills the design goals. In particular, it facilitates the combination of multiple types of information, scales to large deployments as the DDS middleware does, and remains open so that new sensors and logic can be added as the system and technology evolves.

Questions?