

Resource Monitoring for High Availability

Paper Number 4085

Manh Ha
Hewlett-Packard
19410 Homestead Road
Cupertino, CA 95014
408-725-8900

Abstract

There are many components involved in achieving high availability. Careful and thorough planning is one of the major steps to eliminate a single point of failure. Redundant hardware will provide backups in the case of failures. Robust software and highly available packages will decrease outage time. Clustering ensures continuity of services when there is a node failure.

All these endeavors center around the idea of making sure a resource is highly available. There are not many resources that are 100% failure-proof. So the first step is to detect errors. The key to error detection is to keep track of the health of a resource. If and when a failure is detected, recovery action can then be taken.

This paper discusses a framework to implement a resource monitor to react to changes in system and or network resources.

Resource Monitoring

The idea of availability is part of everyday thinking. The phone is expected to function, electricity is expected to be available, and computers are expected to provide services. Any loss of services, planned or unplanned, is known as an outage. The outage is expected to have minimal effect on the users of the services. The goal of a service provider is to avoid or at least minimize the loss of service by making a system highly available.

There are many issues involved to ensure the high availability of a system. A software package can fail and a piece of hardware may stop functioning. Recovery steps must be in place to render the system available should such a failure occur. Hence to ensure that a system is highly available, it is important to verify the hardware and software that the system relies on through monitoring. A model to monitor resources will be discussed.

Components of Resource Monitoring

Resources

In the abstract sense, a resource is any entity that provides services. There are hardware and software components in a system. A LAN interface is a resource. The CPU, memory, and data-storage media are all resources.

A mail-server process can be a resource. By the same token data bases fall into the same category. As a matter of fact, any application which is servicing others can be considered as such.

But a resource is not limited to a piece of hardware OR software. It can be an aggregate. It can be comprised of two LAN interfaces, or three applications. It can contain both hardware and software components. For example: A modem is receiving data through an X.25 connection; the data is processed by application A; this application then pipes it to application B, which displays it. The modem, and applications A and B are each a resource, but they can also be considered as a whole to be a partition of resources.

It is important to note that resources, aggregate or not, are entirely defined by a user. They can vary from system to system. The determining factor is that a resource fulfills a need in terms of creating a highly available environment. The goal is to avoid the loss of service by reducing or managing failures so that the service is available when needed.

Resource Monitor

All the resources in the world will not ensure high availability if there is no mechanism to keep track of them and recover when errors are detected. A monitor can keep track of the health and optionally take remedial actions.

Different approaches can be taken to monitor resources: active monitoring, passive monitoring, and a combination of both. Each of these offers their own advantages and shortcomings. Effort and time to implement are other factors in determining which method to choose.

An active monitor may on its own poll its resources to make sure they are healthy. A LAN monitor may send out a loopback packet through the interface. The goal is to ensure that the interface is capable of sending and receiving data. If the packet is not received, it may try to send out a second or third time. When the pre-defined retry count exhausts, the monitor will consider the interface down.

Resource Monitoring for High Availability
4085-2

Passive monitoring may just be a process waiting for an event to happen. In the case of a LAN monitor, one implementation may be to have the monitor not actively involved in polling the LAN interface. The card will send an error notification to the monitor when it senses a failure.

Another approach may involve augmenting the passive mechanism with certain active maneuvers. Every so often a passive monitor may query the LAN interface to ensure it is functional. Or it may look at the traffic counts to determine whether packets are getting through.

Once an error situation is identified, the monitor will need to take actions. One action may involve trying to recover if the monitor is built with this capability. Another may be to notify the user or another application about this failure. The user or application may then alleviate the situation. Whether the recovery mechanism is a built-in function of the monitor is entirely at the discretion of the developer.

The next important issue to consider is high availability of the monitor itself. Since the monitor is one of the cornerstones of high availability, it is detrimental if for one reason or another it disappears. Its non-existence will provide a false sense of security. Since it is no longer operating, errors will not be detected in a timely fashion and a user of the resources will not be aware of their impending demise.

Client

A resource monitor provides a monitoring service. The entity that requires and initiates this service is the client. A client may need to find out what kind of resources are available on the system and whether monitors exist for each one of them. It also needs to know what type of value a monitor can provide to indicate the health of a resource. A resource can have an UP, DOWN, SUSPEND, or other state. These states are resource specific.

Once it has determined what monitors are available on a system, the client can then start the monitor. The client can tell the monitor whom to send a notification to when a noteworthy event has happened. It can also obtain status information through the monitor and stop the monitor when the monitoring service is no longer required.

Target

Once it discovers that the resource has attained a state that its client has established, the monitor will report the event. It will send the notification to the target that has been specified by the client when the monitor request is made. This target can be the client itself.

One functionality of the target is to take recovery actions if the monitor has not done so. Whether either the target or the monitor takes remedial actions is entirely implementation dependent. The recovery can be programmatic, manual, or a combination of both.

Resource Dictionary

A system usually has more than one resource. The names of all these resources need to be stored in a central location for easy identification. There are also monitors for these resources. Their names are to

be represented in this location. Then an association must be made between the resources and their respective monitors. A resource dictionary serves these functions.

The dictionary can be a data base or flat file. Its format is implementation specific. Some of the goals are fast access time and easy implementation.

This dictionary is to be interpreted by the clients who are interested in learning about resources and monitors. The clients can then determine what monitors, if any, to start. Hence this dictionary is accessed by all the clients. A common interface will not only speed up the development time for an application writer but also isolate users from the specifics of the underlying resources. This leads us to the following discussion.

Registrar

Now that the dictionary contains the names of the resources and monitors, it is desirable to have a mechanism to interpret its content. This mechanism is an expert on the format and semantics of the dictionary. It will provide a common set of interfaces so that each application client does not have to concern itself with the dictionary. This mechanism can be called a registrar.

Furthermore, since it has extensive knowledge of the dictionary which identifies and associates monitors with resources, the registrar can perform other useful functions on behalf of a client. It can start the monitor for the application. It can stop it per the request of the client. It can query the status of the monitor. These services will reduce the development time of all the clients because it offers a common set of interfaces.

Illustration of a Resource Monitoring Model

So far we have covered several components of a resource monitoring model:

- Resource - Service provider
- Monitor - Keep track of the health of a resource
- Dictionary - Associate resources with monitors
- Registrar - Interpret the dictionary, start, stop, and query the monitor
- Client Application - Interested in starting the monitor to track resources
- Target Application - Receives event notifications from the monitor

We will fit the various components into a drawing.

Resource Monitoring for High Availability
4085-4

Figure 1: Resource monitor model

Client
Application

Target
Application

Registrar

Monitor

Dictionary

Resource

The dictionary is the repository for the names of resources and their associated monitors. The registrar serves as a point of contact for the dictionary. The client obtains information on the resources through the registrar. The client can then decide to initiate the monitoring process through the registrar. The monitor starts to keep track of the health of the resource. When a noteworthy event happens, the monitor will send the notification of such an event to the target.

Event Monitoring Service

Now that we have examined some of the components of a resource monitoring model and discussed their roles, relationship, and interactions, the question is “does an application have to develop all the pieces of the architecture?”.

Products such as Hewlett-Packard’s Event Monitoring Service(EMS) provide a framework to facilitate development of resource monitors. It offers end-users the flexibility to devise a more highly available solution for their environments.

First, the components of EMS will be discussed. Then, its features will be presented. Illustrations will be given to capture its architecture and environments.

Components of EMS

EMS is comprised of a registrar, resource dictionary, set of APIs, graphical user interface(GUI), developers’ kit, and Hewlett-Packard-supplied monitors. The registrar and resource dictionary,

Resource Monitoring for High Availability
4085-5

together with the roles they play, have been described earlier. The remaining components will be the subject of the next discussions.

APIs

EMS provides a set of APIs for the client and monitor to establish a highly available environment. The APIs present a consistent interface independent of the underlying resources. Hence end-users are isolated from the specifics of the resources.

Some of the functionalities performed by these APIs are:

- To allow the client to find out the names of resources on a system
- To start a monitor to keep track of a resource
- To set up how often should a monitor checks its resource to ensure the latter's health
- To set up the monitoring criteria
- To specify the target(recipient) of a notification should such a criteria be met
- To terminate the services of a monitor
- To gather status information on the resources being monitored

The APIs can be broken into four categories: general functions, client-specific functions, monitor-specific functions, and target-specific functions.

Graphical User Interface(GUI)

The GUI is a complement to the set of APIs. It provides the end-users an interface to look at the names of the resources. It accepts input from end-users to start/stop monitors. It allows the end-users to specify targets of notification. The GUI behaves like a client of the monitors.

Developer's Kit

The developer's kit contains monitor APIs(including general functions, monitor-specific functions, and target-specific functions), programmer's guide, sample monitor, test tools, and GUI. The monitor APIs are to help application developers implement monitors to ensure their applications are more highly available either in a single or multi-system configuration. The programmer's guide documents the APIs. The sample monitor gives an example of how a monitor can be written, and the test tools ensure robustness of the monitor once it is written.

Now that the various components of EMS have been discussed. It is time to present two drawings to better illustrate the high-availability architectures it can fit in.

Resource Monitoring for High Availability
4085-6

Figure 2: Single-system EMS

In this environment, all the EMS components are contained within one system.

Client Application

Target Application

EMS API

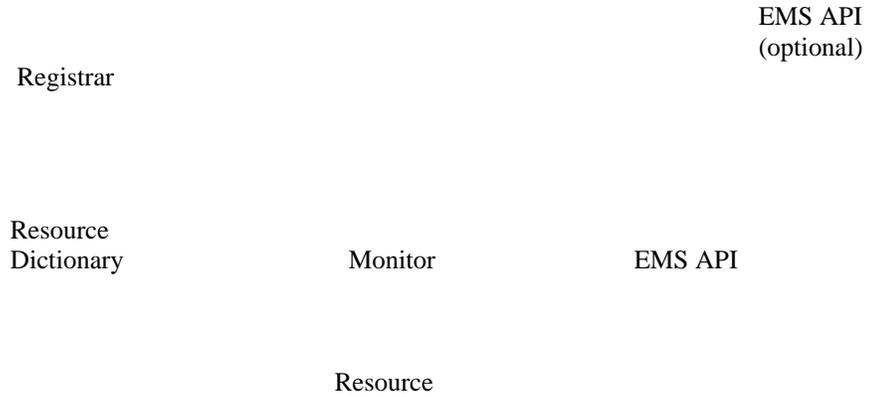
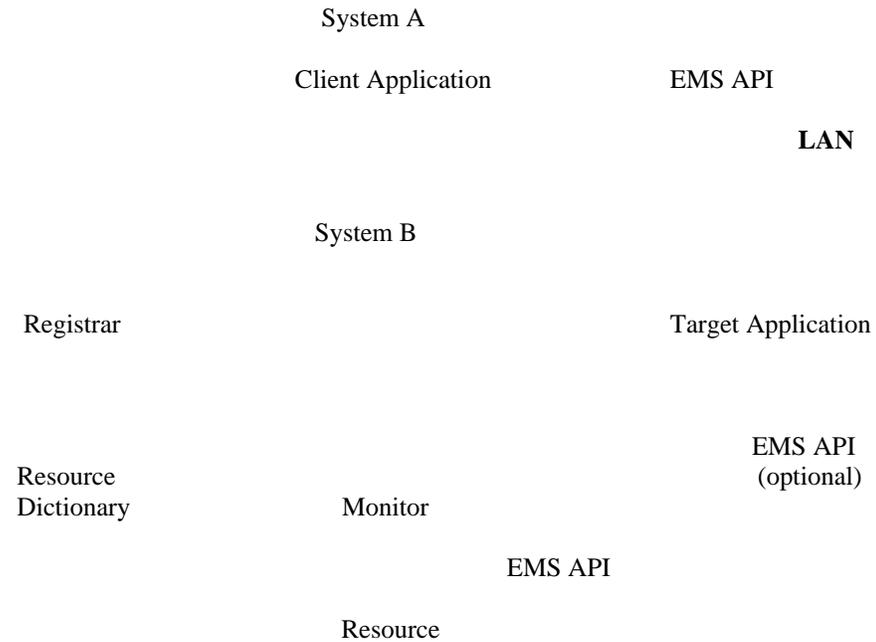


Figure 3: Multi-system EMS

In this environment, the client is on a different node.



Resource Monitoring for High Availability
4085-7

Figure 2 demonstrates a configuration where all the EMS components reside in the same node, whereas figure 3 shows a distributed configuration. The target can be in a node other than the monitor's.

Features of EMS

Now we will examine some features of EMS. The status of a resource can be more than just UP or DOWN. There can be arbitrary status changes, such as when a threshold is crossed. EMS allows the monitor developers to define an arbitrary status for the resources. Then when the monitor is launched,

the application client can tell the monitor to track this status. Once the status changes, the monitor will send a notification to the target. There may be cases where constant notifications of status are desirable. In this case, EMS provides a mechanism for the monitor to send out periodic reports.

Notification to the target can be done in one of several methods. EMS provides the choices of TCP, UDP, SNMP, Hewlett-Packard's MC/ServiceGuard and IT/O messages. This allows a tailor-made monitor for specific environments.

The monitor, client, and targets can reside in a single system, or they can be distributed across a local area network. This provides flexibility in implementation for distributed network configurations. There can be multiple targets for event notification allowing an application developer to be able to deal with an error condition more efficiently.

The monitor has the responsibility to keep track of the health of a resource. Its well being is of primary concern. It is expected that the monitor should be functional at all times to fulfill its duty. Its demise can be disastrous to a high-availability environment. EMS has a built in persistence mechanism to keep track of the existence of monitors. If a monitor is terminated for reasons other than through the proper channel, EMS will restart the monitor. A system may cease to operate as a whole due to one reason or another. When the system goes down, there may be monitors running. It is very desirable for the monitors to resume their operations when the system is up again. The EMS's persistence mechanism will restart a monitor after the system is rebooted.

APPLYING EMS

Application writers can take advantage of the EMS infrastructure to create resource monitors to operate in their environment. Because of the flexibility of EMS, monitors can be written to meet a variety of specific requirements. An application can make use of EMS to make a resource more highly available in a standalone configuration.

Applications writers can also write resource monitors to integrate into Hewlett-Packard's MC/ServiceGuard environment. MC/ServiceGuard provides a highly available solution within a cluster. A resource can be defined as a dependency for MC/ServiceGuard. The monitor will notify MC/ServiceGuard that an event has happened. The latter can then take appropriate actions.

Resource Monitoring for High Availability
4085-8

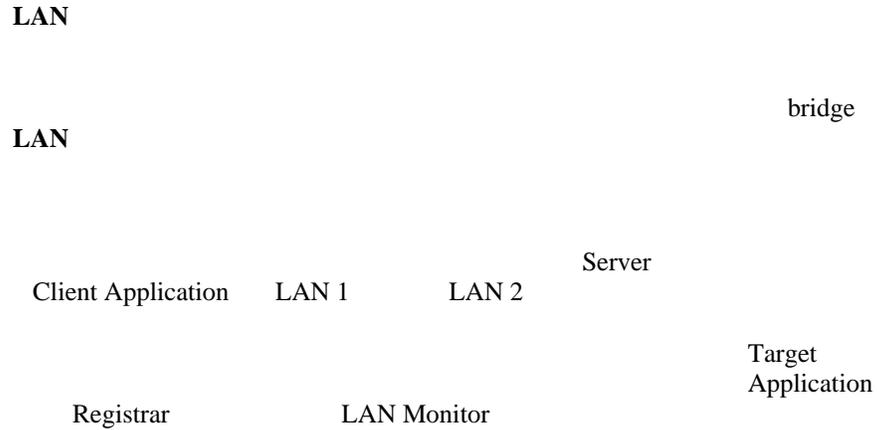
Hewlett-Packard has implemented OTS and ATM monitors. Disk and LAN monitors are under development at this point.

We will describe two situations in which EMS will render a resource more highly available. The requirements and environment are purposely chosen to be simple to illustrate the point.

Figure 4: LAN Monitor

User A

User B

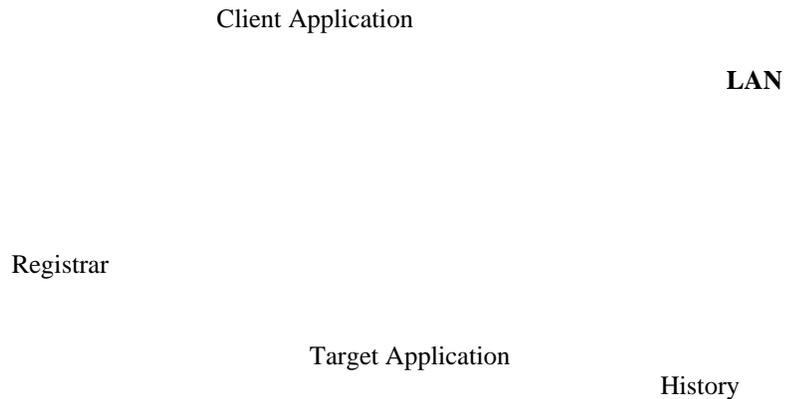


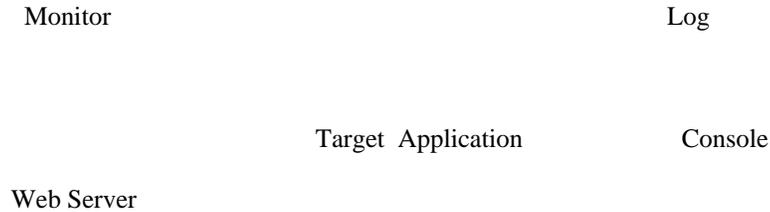
The purpose of figure 4 is to show the functionalities of the LAN monitor. For the sake of simplicity, the resource dictionary is not shown. The scenario provides a high availability solution in case of a LAN or cable failure. LAN 1 is the primary LAN, LAN 2 is the standby. Users normally would log on to the server through LAN 1. The configurations of user systems A and B are not shown, but they are the mirror of the server.

The monitor keeps track of both LANs. If LAN 1 fails or the connection from LAN 1 to the cable is disrupted, the monitor will sense the problem and perform a local switchover of the IP address from LAN 1 to the standby LAN 2. After the switchover, all LAN traffic will go through LAN 2. Applications and users who use a connection based protocol such as TCP, or even a connectionless protocol like UDP with a standard data transmission verification algorithm, will barely notice the failure other than maybe a glitch in the network. The monitor will also send a message to the target application, which can log an error and alert the operator. The operator can then take appropriate actions to determine and fix the problem.

Resource Monitoring for High Availability
4085-9

Figure 5: Web Server Monitor





In this scenario, the sole purpose of the system is a web server. The monitor serves two functions. It ensures the presence of the server process. If for any reason the process ceases to exist, the monitor will restart it and send a notification to the console. The second task of the monitor is to track the load of the server. If a load of 85% is defined as the upper limit, then every time 85% is attained, the monitor will send a notification to the target application, which in turn will log it to a file. This file can be retrieved by the system administrator to make upgrade plans. The picture shows two targets to receive the notifications, but one can handle both situations. The purpose of using the two targets is to demonstrate that EMS can send notification messages to multiple entities.

Further Considerations for Resource Monitoring

Since the resource monitor is one of the basic building blocks of high availability, the monitor must be a trusted process. Trusted in this context means reliable. It must be robust in dealing with any and all error conditions. It must be able to tackle unexpected events that it may encounter while dealing with its resources. It must not hang or terminate on its own. It should be created to be as bullet-proof as possible. Other issues to be considered include considerations for speed, performance, real-time process, memory resident, and pre-allocated memory.

Resource Monitoring for High Availability
4085-10

To emphasize the importance of recovery, we will reiterate some points regarding recovery in a highly available environment. The monitor is the first to detect an error condition; it follows that for performance reasons the monitor is the best candidate to try to remedy the situation. Hence recovery should be part of the functionality of the monitor. But in certain cases, it is possible that the target of the notification is the entity which is more capable of handling the recovery. In extreme cases there may be situations in which the only recovery mechanism is through human intervention. The recovery mechanism, if any, is entirely implementation dependent and it can differ case by case.

It takes careful planning, investigation, coding, and testing. But once a robust monitor is set up, it can serve a highly available environment well.

Resource Monitoring for High Availability
4085-11