Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

# A practical approach to monitoring network redundancy

**Richard Phillips[a], Kouroush Jenab[a*] and Saeid Moslehpour[b]**

*[a]Department of Engineering and Technology Management, Morehead State University, Morehead, KY, USA*
*[b]College of Engineering, Hartford University, West Hartford, CT, USA*

**C H R O N I C L E**

*Article history*:
Received: July 18, 2018
Received in revised format: July 28, 2019
Accepted: September 19, 2019
Available online: September 19, 2019

*Keywords:*
*Practical Network Monitoring*
*SNMP*
*Network Monitoring*
*Network Performance*
*Network Alerts*
*Interface Redundancy Monitoring*

**A B S T R A C T**

Computer TCP/IP networks are becoming critical in all aspects of life. As computer networks continue to improve, the levels of redundancy continue to increase. Modern network redundancy features can be complex and expensive. This leads to misconfiguration of the redundancy features. Monitoring everything is not always practical. Some redundancy features are easy to detect while others are more difficult. It is common for redundancy features to fail or contribute to a failure scenario. Incorrectly configured redundancy will lead to network downtime when the network is supposed to be redundant. This presents a false sense of security to the network operators and administrators. This research will present two scenarios that are commonly left unmonitored and look at a practical way to deploy solutions to these two scenarios in such a way that the network uptime can be improved. Implementing a practical approach to monitor and mitigate these types of failures allows costs spent on redundancy to increase uptime, and thus increase overall quality that is critical to a modern digital company.

## 1. Introduction

A report in 2015 stated that the average adult spends 8 hours and 21 minutes sleeping per day while that same adult spends 8 hours and 41 minutes on media devices (Davies, 2015). All these connected media devices use many forms of networking. We spend more time on networked devices then we spend sleeping. Clearly, computer networks have become critical to many modern-day activities. As the level of criticality has increased, so has the desire to build networks with higher levels of resiliency and redundancy (Bayrak & Brabowski, 2006). Computer network operators that consider the two words similar, typically apply the same types of monitoring for components regardless of their role. Both methods introduce scenarios that can be hard to monitor. This approach will explain methods that will establish monitoring for common redundancy and resiliency features or components. The first study will look at redundancy in the device and the second study will look at redundancy in the connectivity between devices. These are two common scenarios for both simple and complex networks. After establishing a practical network monitoring strategy, the system can detect and possibly repair issues, leading to better

* Corresponding author. Tel: 1(606)783-9339, Fax: 1(606)783-5030
E-mail address: k.jenab@moreheadstate.edu (K. Jenab)

doi: 10.5267/j.ijdns.2019.9.004

uptime metrics. It is common for an organization or company to invest in more redundancy and resiliency features. These configurations can add exponential costs to the design, only to see failures continue and uptime turn into downtime. Using a more practical approach can keep the alerts from becoming background noise and focus monitoring efforts to work in conjunction with the redundant and resilient designs.

## 2. Monitoring overview

The most common form of network monitoring today entails the SNMP protocol. SNMP, or simple network management protocol, comes in both "push" and "pull" varieties. When a network monitoring system polls a device for specific information, it is "pulling" data from the device to the monitoring system. The reverse or "pushing" is when the device sends SNMP data to the monitoring system when a specific event happens. This is typically associated with problems where pulling data on a regular schedule induces delays in detecting problems. The SNMP protocol uses a complex structure that must be understood. A MIB, or management information database, is the structure used in SNMP (Netak & Kiwelekar, 2006). A MIB is setup like a tree with a root and branches. Each branch has specific object identifiers (OID). Most monitoring systems have commonly used MIB trees and are setup to detect the polled device, so it uses the correct OID for polling a basic set of things about that device. These MIB/OIDs have some common branches but most are device manufacturer specific. This leads to large variability in the branches. To setup a network with 10 devices, it is easy to determine what device manufacture MIB you are using and select the correct OIDs for your device. This is optimal but not practical or even typical. Large networks can be far more difficult to establish the correct MIB/OIDs that need monitoring. Most network monitoring system have defaults established to ease initial deployment and then the network operation must customize to the environment. This customization becomes complex and time consuming for large networks. Often overlooked are the redundancy/resiliency features or components. It is not difficult to monitor a single interface or chassis for things like throughput (in bps or input bits per second), link utilization percentage or errors.  As stated previously, SNMP is mostly different for each manufacturer. This creates opportunity for an OID specifically for a second redundant power supply to be hard to derive, which in turn leads to a lack of native polling for that component. If the monitoring system does not handle this out of the box and the network operator does not specifically deal with a failure notification of the power supplies, then the network is at great risk of having a failure that goes undetected. Building on that scenario of a power supply that is purposed for failover, fails itself and goes unnoticed so the next failure takes down the whole device and possibly the network. In summary, the problem will present itself as a complete loss of power when in fact, one of the power modules failed prior to the actual significant event but went undetected.

Another method to monitor devices is using syslog messages, or system logged messages (Gerhards, 2009). Machine data is also a commonly used name for these log messages. Syslog logfiles have been around for years. This data comes in all formats and detail levels. It is in ASCII text and has no real structure. Newer logging systems have become increasingly good at creating intelligence that can add structure to this unstructured machine data and produce valuable information that can be correlated across multiple machines or services. Using this machine data can be a powerful way to solve problems associated with traditional SNMP polled monitoring solutions. Machine logs, when used to diagnose a system, are records of system events that provide details of what the device is doing internally. With the modern machine learning and big-data analytics, mining and analyzing this unstructured data moves traditional reactive support models to more proactive models of support.

Solarwinds Orion (SolarWinds Worldwide, LLC.) is the tool that performed the SNMP polling for this approach. This network monitoring system is multifunctional and has many features. The system can make network device configuration changes as part of an alert remediation strategy. This is a feature that is part of a "self-healing" process (Quattrociocchi et al., 2014). Solarwinds Orion will also collect machine syslog data for some event and alert notifications but this approach utilized Splunk (Splunk Inc) to analyze the machine data used to build the algorithms for failure alerting and remediation. Network alarms need to evaluate and raise events to a status that notifies the network operators. The sheer volume of alarms that can exist on a large network can overwhelm a network operations center. Tuning the thresholds that trigger alarms are part of any practical strategy for network management. The tuning process involves looking at the lifecycle of the alarms and looking at the time the alarm stays in the various states like active, cleared or ended. It a report released in 2009, 20% of network alarms automatically cleared in less than 5 min while 47% automatically ended within 5 minutes (Wallin, 2009). These portions of the lifecycle suggest filtering these alarms out. It is important to consider alarms management when implementing any strategy for monitoring network redundancy. Alarms fatigue should always be avoided if possible.

## 3. Monitoring redundancy and resiliency

Merriam-Webster defines redundancy as “serving as a duplicate for preventing failure of an entire system upon failure of a single component”. Merriam-Webster also defines resiliency as “tending to recover from or adjust easily to misfortune or change” (Merriam-Webster.com). Clearly the definitions are different, yet they are often thought to mean the same thing. This can lead to a lack of adequate alerting or monitoring. A detailed study of each term, as it relates to computer networks, will help to better understand the approach and how each complement one another when maintaining a redundant and resilient network design.

### *3.1 Redundant*

A network designed for redundancy has multiple components that prevent the device from becoming unavailable or down. This covers things like power supplies, additional CPUs, hard disks or memory modules. A typical example of redundancy in a network device is a device that has multiple power supplies or power modules. Having multiple power supplies connected to multiple power sources eliminates outages if you lose primary power. While this does make the system resilient, the true definition is closer to redundancy. When the redundant power supply fails while its running on the primary power, the network operator must be notified of event. If the failure goes unnoticed, a primary power supply failure will create a network down event for the entire device. When SNMP monitoring a network device for basic things, like is the whole device up or down, unless configured specifically, there will not typically be a down message until both power supplies fail and the unit becomes unreachable. When monitoring a redundant system, we must make sure to raise alerts when the redundant device component fails or is in an unusable or degraded state. Typically, the machine data is a better way to predict or identify these types of problems compared to simple SNMP polling. The machine data has a greater level of detail that can be leveraged in a modern log collection system. Some operators utilize SNMP traps, or “pushes” for this time-sensitive event notification.

### *3.2 Resilient*

Resiliency in a network device means the device or circuit can stay online and working during a failure. In some computer networks, this is the ability to route around problems using multiple paths to the same destination. This could be defined as a form of self-healing (Quattrociocchi et al., 2014). Another form a resiliency is to build multiple paths to a logical device, made up of two physical devices, and bundle them all together, so they look like a single connection. Fig 1 is a visual representation of port channel technology with 2 or 4-member interfaces. Looking at Fig 1, specifically when looking at Po2 on the left side, there are 2 circuits in this port channel. The extra circuit serves as a redundancy measure as well as a resiliency measure. On larger port channels there could be 4, 8 or even 16 circuits (as shown on Fig 1 - right). If a member circuit develops problems and goes offline, the remaining circuits continue to pass data. In the case of a larger port channel it may not be noticeable to the users or network operator from a consumed bandwidth perspective. Therefore, alerting is critical. For example, hypothetically if Te4/1 on either side of Fig 1 experiences problems but continues to pass data, the data may have errors or discards on the transmit or receive side of the circuit. These types of issues create delays for customers or they slow down the communications far beyond expected rates. While the port channel is still operational and will appear to be up and resilient, the intended design is not what the customer experiences. It would seem logical that the other 7 circuits should handle the traffic and automatically exclude the defective member. Traditional network devices do not have circuit error detection or self-healing features. Also, there are times when errors are normal such as a denial of service attacks or a defective machine might flood the network with unintended packets of data. To properly fail over these circuits, removal of the defective member from the port channel must occur. If the monitored interface is only detecting up or down status, the network operator is not aware of the problem when it occurs. This scenario, where the network operator does not know there are problems on the network, yet the customers are experiencing no problem, is the opposite of what resiliency is supposed to accomplish for a network.

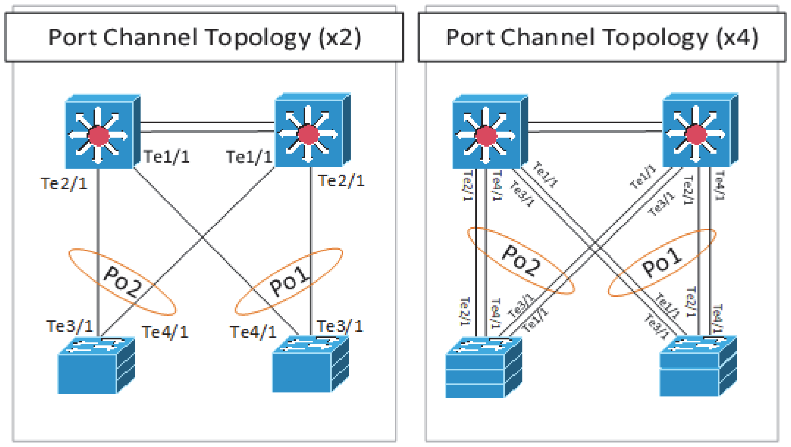
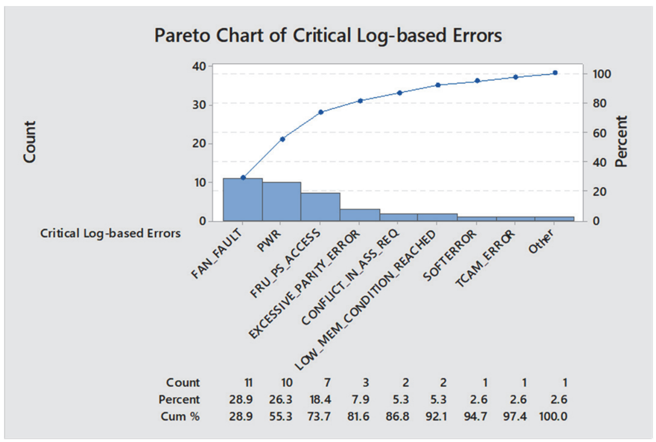


**Fig. 1.** A visual representation of port channel technology

## 4. Research

The first case study looked at a year's worth of log data from a large university network. The network is comprised of over 2500 physical devices. The logs are captured in a log collection system. A list and count of critical events was derived from a report obtained from the log entries in the log collection system. A pareto chart with the event data was used for evaluation of the most common critical log messages See Fig 2. Overall there were 38 critical log messages for the time period studied. The three most common were fan failures and power supply issues. After investigation of the power supply issues; nearly 46% (7 of 17) of the failures occurred on redundant power supplies (FRU_PS_ACCESS & PWR).



| Critical Log-based Errors | FAN_FAULT | PWR | FRU_PS_ACCESS | EXCESSIVE_PARITY_ERROR | CONFLICT_IN_ASS_REQ | LOW_MEM_CONDITION_REACHED | SOFTERROR | TCAM_ERROR | Other |
|---|---|---|---|---|---|---|---|---|---|
| Count | 11 | 10 | 7 | 3 | 2 | 2 | 1 | 1 | 1 |
| Percent | 28.9 | 26.3 | 18.4 | 7.9 | 5.3 | 5.3 | 2.6 | 2.6 | 2.6 |
| Cum % | 28.9 | 55.3 | 73.7 | 81.6 | 86.8 | 92.1 | 94.7 | 97.4 | 100.0 |

**Fig. 2.** A pareto chart with the event data

In this scenario, the SNMP-based network monitoring system did not detect an event that was critical or problematic for the failed power-related components. The failures were reported by customers and added to the incident management system. Technicians were dispatched to repair the failure and restore the customer experience. In the case of the redundant power supply failures, an algorithm was added to the Splunk analytics engine to process the log data to detect redundant power module failures. The system was also configured to automatically open incident tickets for further technical investigation for detected events. Post process implementation, the network devices can experience power supply failures that would not normally be detected during SNMP polling but are still remediated using log data. This allows for proper network operator alerting and problem remediation. In summary, this new workflow has allowed the devices that experience failures with redundant components to be repaired without causing downtime for the users. The 7 failures that occurred with a redundant component more than likely created downtime on a redundantly designed device. These failures could have been prevented but the failures largely went unnoticed until the system experienced a total power failure.

The second case study looked at the same university network for port channel usage on the campus network. This university has over 4000 fiber optic connections and many of them are members in port channels. Root causes of downtime events suggest that these interfaces go bad without notification or the notification is lost in an overwhelming volume of alerts. This issue was reported by the network operator prior to the study and suggested it warranted a long-term solution. The issue is that the most important tiers of the network, core and distribution, have numerous port channels to allow for capacity and resiliency. These port channels lose members due external issues like fiber trouble and begin to throw errors when passing traffic. The individual interface is generating errors, but the circuit never goes completely down. It is just very slow for traffic flows utilizing that port channel member since the traffic is resent several times due to errors. This creates a bad user experience. The study identified core and distribution level port channels. After identification, the network monitoring system implemented custom tags, attached to the specific interface, to categorize these interfaces for event detection. After tagging, an event detection algorithm was created to capture this segment of interfaces if they generated any of the specific errors in Table I. These types of errors are indicative of a member that should be removed from a port channel and investigated or repaired (Wallin, 2009).

**Table 1**
Specific errors

| Error type | Duration | Typical Causes |
|---|---|---|
| Input Errors (RX) | Last hour | Fiber or cabling Inconsistencies, Malformed Packets, Speed or duplex mismatch, Queue or Buffer Flooding |
| Input Errors (RX) | Today | Fiber or cabling Inconsistencies, Malformed Packets, Speed or duplex mismatch, Queue or Buffer Flooding |
| Output Errors (TX) | Last Hour | Flow Control Problem, Fiber or cabling Inconsistencies, Malformed Packets, Speed or duplex mismatch, Queue or Buffer Flooding |
| Output Errors (TX) | Today | Flow Control Problem, Fiber or cabling Inconsistencies, Malformed Packets, Speed or duplex mismatch, Queue or Buffer Flooding |

The detected events generated both alarms and actions. The alarms logged the event in the event list while the actions used the device information to remove the member from the port channel. This was properly logged in the event list and an email was sent to the network operator queues for notification of the issue along with remediation steps taken. The network operators noted over 300 interfaces that met this criterion. After further review, the event thresholds were adjusted to look for the high probability offenders. It was noted that some interfaces generate errors when the switch buffers get full or the forwarding queues get full. This is not a defective interface but rather the traffic is either too great or there are too many packets per second to process. This requires traffic engineering to engage and pinpoint the solution. The study excluded these types of errors. Likewise, if a fiber is damaged but not severed, the error counts will grow every hour in large numbers and after two polling periods detection of the problem can occur in a process that is in control. This is where the study focused on an acceptable approach.

## 5. Methodology

These studies implemented specific solution sets using specific products. The approach will work with most enterprise class monitoring software systems. There are 5 basic steps required in this approach. The steps are slightly different if hardware replacement is normal for the resolution of the problem (aka power supply failures). These differing workflows are in Fig 3.

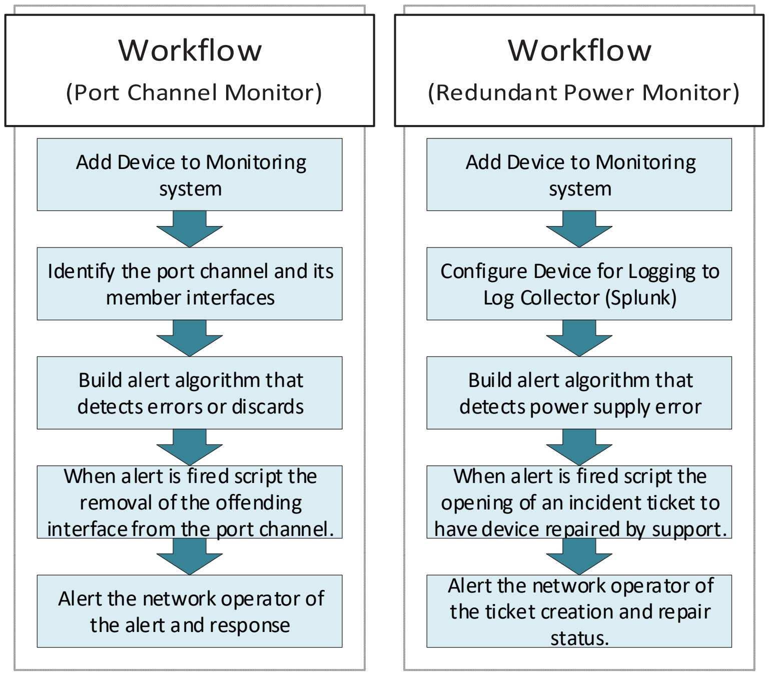

Workflow
(Port Channel Monitor)
Add Device to Monitoring system
Identify the port channel and its member interfaces
Build alert algorithm that detects errors or discards
When alert is fired script the removal of the offending interface from the port channel.
Alert the network operator of the alert and response
Workflow
(Redundant Power Monitor)
Add Device to Monitoring system
Configure Device for Logging to Log Collector (Splunk)
Build alert algorithm that detects power supply error
When alert is fired script the opening of an incident ticket to have device repaired by support.
Alert the network operator of the ticket creation and repair status.

**Fig. 3.** Monitor workflow

Step 1: adding the device to the network management system. This step is straightforward. In adding the device, it is critical that the interface and the port channel be polled for status and error counts. If the network management system does any kind of configuration management including the allowance of script execution, remediation of the event is more than likely possible. If not possible, any form of auto-healing will be difficult. A fallback option would be to alert an operator and continue until the network operator acknowledges the issue. This is typically a standard feature with network management systems.

Step 2: identify the member interfaces of the port channel and segment them out. The network management system can accomplish this step. Solarwinds Orion allows for custom properties for all interfaces and devices (nodes). If that option is not available, renaming the interfaces in the device may allow for segmentation that detection algorithm can use. Either way the interfaces need to be detectable and look different, so they are not like the normal event messages.  In the case of redundant hardware, the machine data needs to be identified from manufacture literature or device testing. Once the log messages are identified they can be placed into the correct location of the algorithm.

Step 3: build alert algorithm that detects errors or discards. When configuring the event/alert, it is critical that the search for member interfaces use the segmentation strategy used in step 2. This will allow for custom notifications or actions that will get through any network alert noise. It is important to look at both errors inbound and outbound. It is not uncommon for a circuit that goes between two monitored systems to experience errors in one direction. For example, if looking for an inbound error on device A (receiving interface), it would seem like there should be an output error on device B (sending interface). This is not the case and if the algorithm is only detecting outbound errors, the process will miss any inbound errors.

Step 4: when alert is fired, script the appropriate action or the alert. In the case of a defective interface the next step may be the removal of the offending interface from the active port channel. This is not always possible depending on the capabilities of the alerting system. While self-healing is possible, it is not always desired. It is more important to raise attention to the issue for mitigation or resolution. Depending on the system it may better to remove the port channel member if it there are more than 1 active members. It is important to note, this will depend on the device, as different vendors and specific models have different operating systems and scripting abilities. For redundant hardware failures the action might be to engage a support team.

Step 5: alert the network operator of the alert and response. Any identified activity that happens should be logged and the details of the workflow should be sent to the network operator for confirmation of problem resolution or mitigation. Any self-healing system can attempt to heal and miss the mark and result in doing harm. Having knowledge that the event occurred is important. In this study one of the alerting actions opened an incident ticket so there was a record in the incident management system. This ticket tracks and documents problem resolution after removal of the member from the port channel. After this process occurs the network operator will move the interface back into the port channel and resume normal operations.

## 6. Conclusion

Redundancy and resiliency are important features to any complex or large network operator. With uptime being a driving metric for complex networks, it is imperative that the monitoring is easy to replicate and understand. Power supply problems are common, partly due to heat and device cooling system failures. Interface members participating in a port channel can fail without notification and they can turn the customer experience upside down. These failures will happen on redundant or resilient components without warning. When these failed components need to be functional for their role in the design, they are incapable of performing their task. Simple monitoring workflows for both scenarios can ensure redundancy components are available when the system needs them most. Most, if not all, of the network monitoring systems and log collection systems developed have the ability or features to implement a practical approach to monitoring redundant and resilient network components. Don't overlook them or your money spent on redundancy and resiliency could cost you even more.

## References

Davies, M. (2015). Average person now spends more time on their phone and laptop than sleeping, study claims. *Daily Mail. com, Retrieved June*, *4*, 2016.

Bayrak, T., & Brabowski, M. R. (2006). Critical infrastructure network evaluation. *Journal of Computer Information Systems*, *46*(3), 67-86.

Gerhards, R. (2009). The syslog protocol.

Netak, L. D., & Kiwelekar, A. W. (2006). Efficient network management using SNMP. *Journal of Network and Systems Management*, *14*(2), 189-194.

Quattrociocchi, W., Caldarelli, G., & Scala, A. (2014). Self-healing networks: redundancy and structure. *PloS one*, *9*(2), e87986.

Redundant, "Merriam-Webster.com" 2018. [Online]. Available: https://www.merriam-webster.com/dictionary/redundant. [Accessed 17 July 2018].

Resilient, "Merriam-Webster.com" 2018. [Online]. Available: https://www.merriam-webster.com/dictionary/resilient. [Accessed 17 July 2018].

SolarWinds Worldwide, LLC., "Solve your toughest IT management problem, today". 21 07 2018. [Online]. Available: https://www.solarwinds.com/.

Splunk Inc, "Turn Machine Data Into Answers". 2018. [Online]. Available: https://www.splunk.com/. [Accessed 21 07 2018].

Wallin, S. (2009). Chasing a definition of "alarm". *Journal of Network and Systems Management*, *17*(4), 457.