Reducing the Backup Window for SAN-based Backups Using Graph Models

EMC Proven™ Professional 2008

Krasimir Miloshev
EMC Corp.
Miloshev_Krasimir@emc.com

Table of Contents

Disclaimer: The views, processes or methodologies published in this compilation are those of the authors. They do not necessarily reflect EMC Corporation's views, processes, or methodologies.

Abstract

This article will help you to optimize backup operations by reducing the total backup time (a.k.a. Backup Window). We will apply an optimization method based on Graph Theory's single-source longest-paths algorithm. We can apply this method by creating a software program called "Backup Job Scheduler." Existing backup programs are based on "manually" entered backup time schedules. The Backup Administrator has to schedule (determine) the starting time of all operations prior to running the backup program. We will use an automated generator to generate the backup schedules and reduce the total backup execution time instead of manually scheduling backup tasks.

We reduce the backup window by reducing the total execution time for the backup operations. Let us assume we must perform a large set of backup jobs of varying duration. Let us also assume we have some precedence constraints set by precedence relationship (one job must be completed before the next can be started).The question is how to minimize the amount of time to complete all the jobs by satisfying all the precedence constraints. We will perform optimization using the well known in Graph Theory (Job Scheduling) and single-source longest-paths problem for acyclic graphs.

Keywords: SAN, backup optimization, backup window, job scheduling, SSLP (single-source longest-paths algorithm), acyclic graphs, Master Server, SAN Media Servers

1. Introduction

We used to have disk arrays connected directly to servers and tape devices prior to SAN within traditional storage architectures (direct attached storage or DAS). The data path started from the server and went to disk arrays and finally to tape for archiving. Both disk and tape devices were tied to the same server, which didn't provide flexibility during the decision-making process.

One of the SAN's goals is to make these disks and tape devices independent from the servers; to be purchased independently, to be tested independently etc. The other issue with the traditional no-SAN architectures is that external disk memory is not used optimally and effectively. There is no flexibility on using the external memory since those units are tied to certain servers.

SAN-based systems use a common storage area consisting of all storage units. This common storage area is called storage pool, and avoids issues such as one server not having enough memory and another server having more than it needs.

The most important reason to implement a SAN is to take the backup/restore traffic out of the LAN/WAN. This will improve LAN/WAN speed and effectiveness. Centralizing and securing data and improving data center manageability are some other reasons for implementing SAN.

Reducing SAN maintenance downtime is critical since we have a large number of back-up/restore operations and a large number of SAN nodes-servers, disk arrays and tape devices. Generally speaking, we plan backup-operations on daily, weekly and monthly bases. We also run backups on demand.

We have the following types of backups:

1. Full backup- once per week.
2. Incremental backup- daily.
3. Differential backup –data changes between the last full backup and some particular day, for instance we make an incremental backup on Wednesday for all the changes happened from the Sunday's full backup on.

Here are some basic terms for the backup/restore operations:

1. Backup Window- this is the time interval for executing backup operations. We do incremental/differential backups on daily basis (off-work hours are the best time) and full backup (during the weekends) on a weekly basis. So there are two types of backup windows-overnight backup window and weekend backup window.

2.Backup data amount-the total data to be backed up.

3. Network Transfer Rate-the amount of backup data transferred per sec. We run different tests to get the rate. Let's assume that P is the number of all parallel backup sessions running, then P=Amount_of_data_for_backup / (backip_window * network_transfer_rate)

2. Exposition

2.1. Architecture of the traditional backup infrastructures

The basic components of the typical backup infrastructures are:

1. **Master server**-the central management and configuration server for the backup/restore operations.

2. **Clients**-these are clients from backup perspective but from functional point of view they are servers –UNIX servers, DB servers, WEB servers etc.

3. **Media servers**-they backup the client's data controlled and managed by the master server.

4. **Storage backup devices**- usually tape devices/ libraries to store data on.

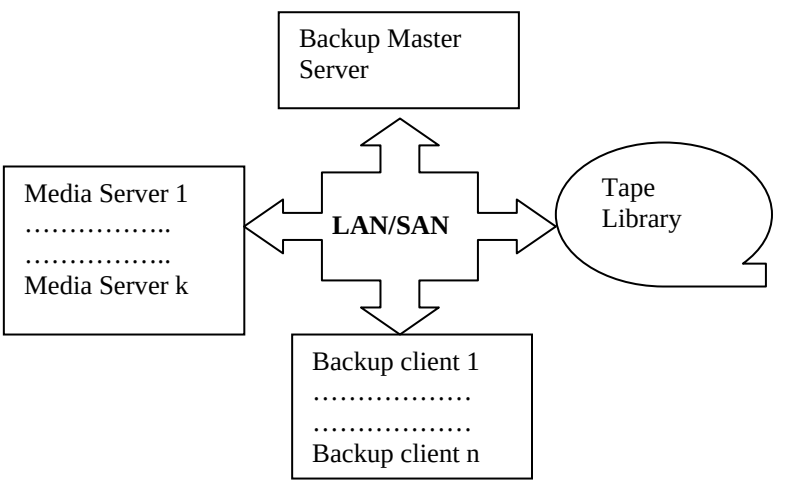

Fig.1

2.2. Architecture of a SAN-based backup infrastructure

Let us introduce a backup infrastructure, based on SAN (storage area network) Media Servers and VTL (virtual disk library) connected to the SAN.VTL is based on tape virtualization over disks. Using a special virtualization software, a number of virtual tapes and virtual tape drives are created instead of using physical tapes. These virtual tapes are implemented over physical disks.

Disks, unlike tapes, are devices with direct access. R/W access parameters are constant and don't depend on the location. Tapes are devices with consecutive access. That's why R/W operations don't have constant parameters. Also, when multiple computers back up across the LAN to a library through "regular" Media Servers, they transfer their data through the LAN network and the backup data stream is just a part of all data streams across the LAN network. So obviously the backup throughput heavily depends on the whole LAN traffic. On the other hand, we create SANs to exclude backup traffic from the entire LAN network traffic.

We have policies and number of clients affiliated with each policy in the "traditional" backup infrastructure. Each policy is affiliated with a particular media Server. So each Media server handles backups for certain groups of clients.
We can try something different.

Let's describe a backup infrastructure, based on Master Server, SAN Media Servers, Virtual Disk Library (based on disks instead of tapes) and no backup clients. The SAN Media Servers backup only themselves. Their licenses are less expensive compare to the standard (regular) Media Servers and are used either to backup large applications (Oracle DB, SQL, Exchange, SAP etc.), or to backup large amount of data (file servers etc.).

We propose a solution based on:

1. Virtual tape Library (disk based) to assure constant data R/W parameters.
2. SAN Media Servers instead of regular Media Servers.

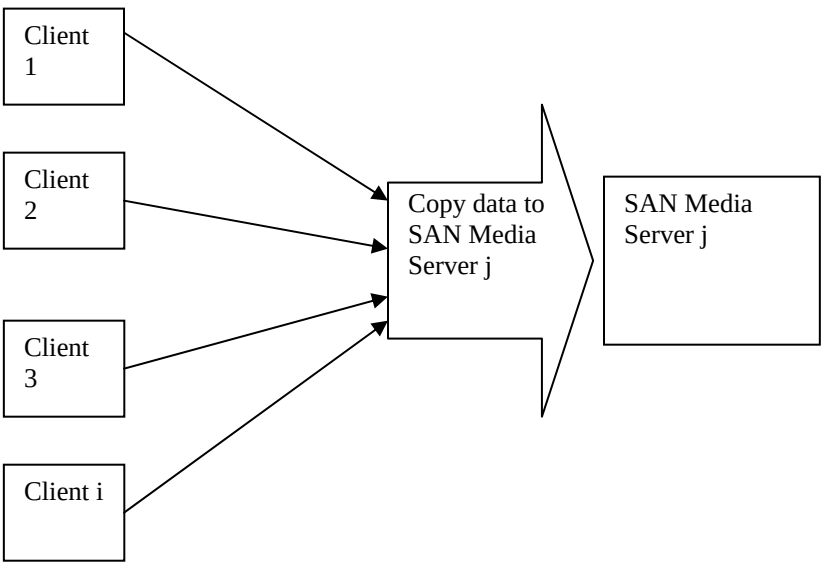

Fig.2

We can copy those small sizes on daily basis to the SAN Media Servers and then the SAN media Servers backing up only themselves during the backup Window instead of backing up relatively small amounts of data on big number of clients (Fig.2). Thus we will have a backup infrastructure consisting only of one Master Server and certain number of SAN Media Servers. No clients, no "regular" Media Servers. Let us show an example.

Let us suppose we have 100 servers. 90 out of them are development "light duty" servers, and the rest 10 are "heavy duty" application servers-Oracle DB, SQL, Exchange etc. In a typical backup infrastructure, we will eventually have one Master Server, 100 backup clients and for instance 2 "regular" Media Servers backing the data of those 100 clients to a tape library.

In our proposed advanced backup infrastructure, we will have 1 Master Server, 0 backup clients, and number of SAN Media Servers, backing up the data to a virtual disk library. How many SAN Media Servers we would need? It's not difficult to determine. We will set up a SAN Media Server- that means 10 SAN Media Servers for each of the 10 "heavy duty" application servers.

We have 90 "light duty" servers left. Since the amount of data to be backed up for each of them is relatively small, we can put 1 SAN Media Server for each 18 of them and that means 5 additional SAN Media Servers. We are supposed to set up 15 SAN Media Servers and one Master Server for the entire backup infrastructure. (Fig. 3)

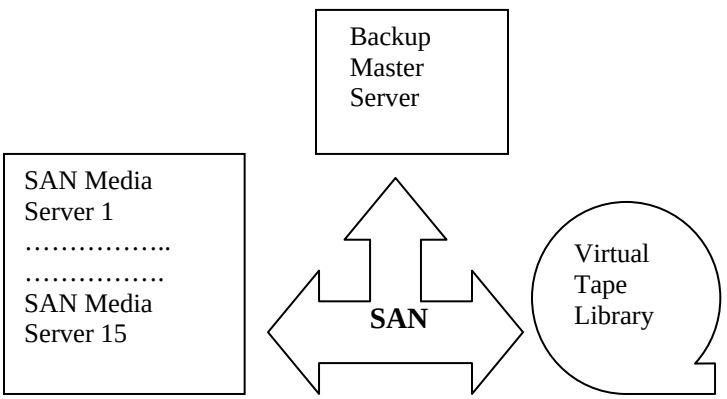

Fig.3

The backup throughput does not depend on the network traffic and becomes more predictable because we use only SAN Media Servers connected to a Virtual Tape Library through a SAN. Backups become much faster as well.

1. We can determine the size of each save set (object to be backed up) in the schedule.

2. To estimate the **data_size** (how much data would be backed up) you can run a command. On Solaris platforms we have the option “s” which returns the size of the file to be backed up: **ufsdump 1fs /dev/rmt/0 file1**

3. We can also determine the backup transfer rate (GB/sec) for each of the servers. The transfer rate for each file server can be accurately determined by running numerous data transfer tests. In SAN, the backup traffic occurs out of the LAN, so backup transfer rate doesn’t depend on LAN speed or LAN traffic. Since all fibre cables are directly connected to the servers, storage devices and the switches, backup transfer rates depend only on specific parameters such as bandwidth of those SAN devices. Thus, they can be determined quite precisely by multiplying the transfer rate and the data size. We can use some well known commands instead of trying to get the transfer rate.

For instance (on UNIX servers) to figure out the backup time **time_backup** for a certain file we can run the following:
dd if=/dev/zero of=/dev/rmt/san_tape_number bs=data_size count=data_size

We run usually full backups during the weekends (Saturday, Sunday), so we are not under such time pressure as during the daily incremental or differential backups. We have a 48-hour backup window for full weekly backups, but a 12-hour maximum backup window for the daily incremental and differential backups. This way **we can estimate the backup duration time in advance** for each of the client’s backup jobs running during the backup Window. That is one of the basic results that we will use in our research.

3. Projecting an automated Backup Jobs Scheduler to minimize the Backup Window in a SAN-based backup infrastructure

Our goals in this research are to optimize backup operations, to reduce the total backup time and to reduce the backup window in a SAN-based advanced backup infrastructure. We can estimate the backup duration for each backup job over that infrastructure in advance. This will be done using an optimization method based on Graph Theory and implementing this method by creating a software program called Backup Job Scheduler.

All existing backup programs are based on manual backup scheduling. The Backup Administrator is supposed to schedule and determine the starting time of all the backup jobs prior to starting the main backup program itself. We will use an automated generator to generate all the backup schedules and reduce the total execution time instead of manually scheduling backup tasks. We reduce the maintenance window and the systems downtime by reducing the total execution time for the backup operations. In this acyclic graph model, each of the backup operations is represented by vertices with weights indicating the amount of time required for certain backup operation to be completed.

priority	backup jobs
1	1
2	2,8,10
3	4,9
4	5,7
5	3,6

Fig.4

Each backup operation can be started as soon as the previous one is complete. Each backup job has own priority, so different jobs have different priority rates that determine which job runs after which other job etc. (Fig.4)

1. Backup jobs 4 and 9 can not be run before 7 and 8;
2. Backup job 8 has to be run after 1;
3. Backup job 10 has to be run before 7 and 5 but after 1;
4. Backup job 3 must be executed after 9;
5. Backup job 5 has to be run after 10;
6. Backup job 6 has to be run after 4;

Also each job has a specific duration in min., so we can present all the data in a table (Fig.5):

Job number	1	2	3	4	5	6	7	8	9	10
Job duration/min.	43	53	51	38	39	46	22	31	32	29

Fig.5

Using a graph model, we would get an acyclic graph (Fig. 6)

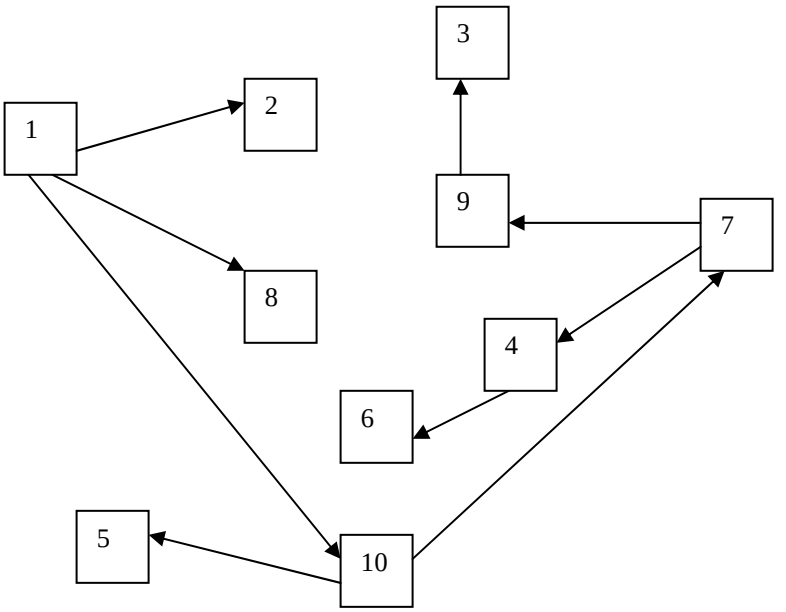

Fig.6

Once again, we are trying to create a method to automate this part of the job and create automatically created schedules. This approach would reduce the total backup time and the Backup Window.

We have to implement a program that generates an automated schedule for each backup operation within the SAN. This is the most important moment as schedules are, to this point, generated manually. Restraints are defined by the backup window.That means both backup start time and backup end time have to be within the window time frame. Our goal is to build a program creating a backup job schedule automatically within the given backup window.

3.1 Job Scheduling problem for backup jobs

We must perform a large set of backup jobs of varying duration. We have some precedence constraints set by precedence relationships (one job must be completed before the next can be started).The question is how to minimize the amount of time to complete all the jobs by satisfying all the precedence constraints. This is the well known **Job Scheduling Problem.**

1. We can determine the size of each save set (object to be backed up). You can run a command to estimate the data_size or how much data would be backed up. For instance, we have the option “s” for Solaris platforms that returns the size of the file to be backed up:

(1) data_size=ufsdump 1fs /dev/rmt/0 file1.

2. We can also determine the backup time for specific backup object (for instance that can be a file). In a SAN, the backup traffic occurs out of the LAN, so backup transfer rate doesn’t depend on such factors like the LAN speed and the LAN traffic. Since all fibre cables are directly connected to the servers, storage devices and the switches, backup transfer rates depend mostly on specific parameters such as bandwidth of those SAN devices. They can be determined quite precisely by multiplying the transfer rate and the data size itself. We can use some well known commands instead of trying to get the transfer rate. For instance, to determine the backup time on UNIX servers time_backup we can run the following command:

(2) backup_time=dd if=/dev/zero of=/dev/rmt/san_tape_number count=data_size

All backups are run during the Backup Window. This is a specified time interval, for instance starting at 6 p.m. and finishing at 6 a.m. the next day. We have a longer backup window for full weekly backups. Based on the considerations above, we concluded that we can estimate the duration time for each backup job in advance.

Our goal is to make a schedule for all backup jobs within the Backup Window. We can collect large sets of statistical data that could vary due to size and starting time variation by running the (2) commands multiple times.

Also, there are different products such as EMC Backup Advisor (EBA) that can collect such statistical data. We have a case of uncertainty due to backup_time variation.

We can solve the Job scheduling problem by using the Critical Path Method. A list of all activities is required to complete the project (also known as Work breakdown structure), the time (duration) that each activity will take to complete, and the dependencies between the activities.

The **Critical Path Method** (**CPM)** is a method to schedule a set of activities. The essential technique for using CPM is to construct a model of the task that includes the following:

1. A list of all activities required to complete the task
2. The activity time (duration) that each activity will take to completion
3. The dependencies between the activities

Using these values, CPM calculates the longest path of planned activities to the end of the entire task, and the earliest and latest that each activity can start and finish without making the task longer. This process determines which activities are "critical", meaning they are on the longest path. The activities on the longest path are called, because any delay on any of those activities would delay the completion of the entire task.

The Program Evaluation Review Technique (PERT) deals with the uncertainty in activity times. In essence, PERT is the same as CRM except that activity times are replaced with expected activity times.

PERT requires that three time estimates be made to calculate the expected time of an activity:

A: the optimistic activity time
B: the realistic activity time
C: the pessimistic activity time

The expected activity time is estimated to be [9]:
expected_time = 1/6 * A + 4/6*B + 1/6 *C

In other words, a weighted average of the three time estimates is taken, where A and C have weight 1/6 and B has weight 4/6.

Let have for backup job A=42 min.; B= 45 min. and C=36 min.

The expected backup time is = 43 min. Thus we can calculate the expected durations for all backup jobs 1..10. We can enter all the expected duration times in a table (see fig.7):

Job number	1	2	3	4	5	6	7	8	9	10
Job duration/min.	43	53	51	38	39	46	22	31	32	29

Fig.7

3.2 Algorithmic approach for resolving the job scheduling problem

From the Graph Theory perspective, the **CPM** can be reduced to implementing the **single-source longest-path algorithm for acyclic graph (SSLP)** (Property 1 and Property 2, pg. 321 on [2]).

Since job 1 is the starting job and all other jobs begin after this job is done, we can present the Fig. 6 graph with a Fig.8 graph. These 2 graphs are functionally equal, but we will now use the edges instead of vertices to present the job durations. The edge (1-2) will represent the duration of job 2, the edge (1-8) will represent the duration of job 8 and (1-10) will represent the duration of job 10. Further one (10-5) will represent job 5 and (10-7) will represent job 7. (7-9) represent job 9 and (7-4) represents job 4. And finally (9-3) represents job 9 and (4-6) represents job 4.

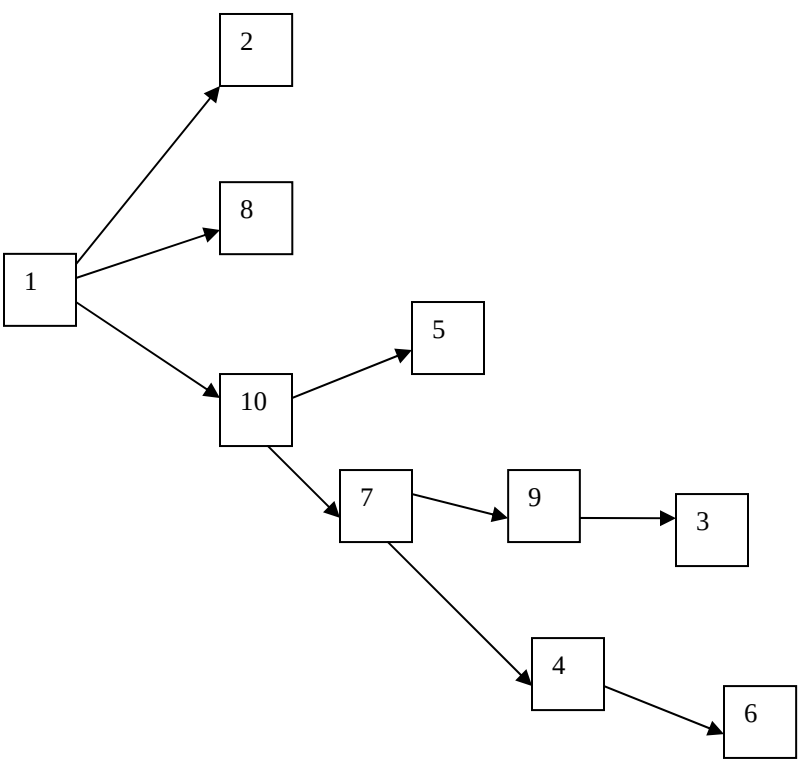

Fig. 8

We can apply a modified Dijkstra's algorithm for finding the longest path in acyclic graph, which was our goal.

Dijkstra's algorithm is used for finding the source-target shortest path in a graph. But it is known (see [6]) that it can be easily modified to find the longest path in a acyclic graph- instead of applying the shortest edge on each step, we will apply it to the longest edge on each step of the Dijkstra's procedure and thus we will get the searched longest path.

4. Conclusion

The critical path we found is: 1-10-7-4-6

In Fig.9 we have presented the Pert diagram itself, showing the jobs sequence. Also we have presented the time/job schedule in min. for each job (Fig.10). The starting time point is 0 and each job starts certain amount of minutes behind that starting point.

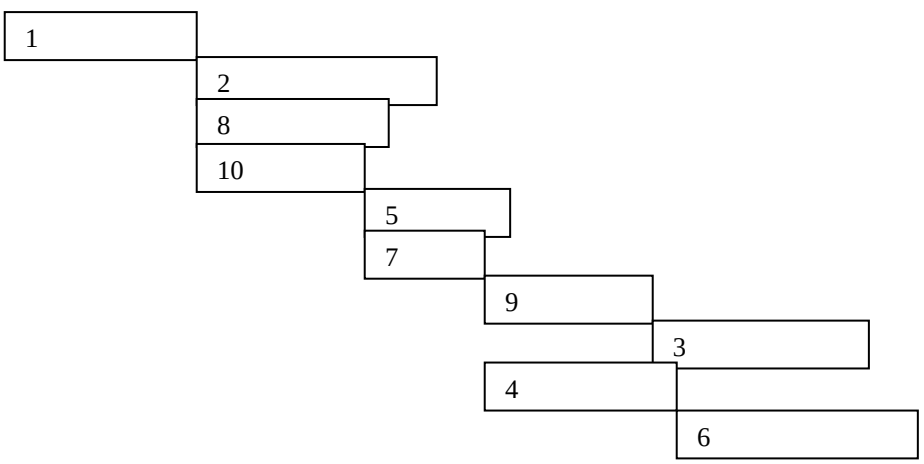

Fig.9

Here is the starting time/job schedule in min. for each job (Fig.10). The starting time point is 0 and each job starts certain amount of minutes behind that starting point.

Job number	1	2	3	4	5	6	7	8	9	10
Starting time	0	43	126	94	72	132	72	43	94	43

Fig.10

So the total amount of time for these automatically scheduled backup jobs would be sum of durations of the following backup job: 1 +10 + 7 + 4 + 6, meaning the total duration would be = 184 min. or 3 hours 4 min. We have run all those backups on a VTL EDL 4400 and we achieved a total backup time of approximately 4 hours 20min.

Experimental results also show that the proposed technique provides a significant time improvement (almost 1 hour 20 or nearly 33 %). In our case, we can reduce the downtime by at least 25-30%, which is significant. The main contribution of this approach is that the Backup Window can be reduced by applying single-source longest-paths algorithm and SAN-based backup advanced infrastructure.

5. References

[1] Wilson, Robin J. Introduction to graph theory, 1983, Addison-Wesley

[2] Sedgewick , Robert Algorithms in C,1990, Addison-Wesley, Boston, MA

[3] Ogletree, Terry William Fundamentals of Storage Area Networks, 2003

[4] Little, David, Chapa, David Implementing Backup and Recovery, 2002, O'Reilly Media, Inc.

[5] Hiles, Andrew, 2003, Business Continuity, Rothstein Associates

[6] Evans, J. and Minieka, E. Optimization Algorithms for Networks and Graphs, 1992, Marcel Dekker, Inc, New York, NY