Modeling Damage Paths and Repairing Objects in Critical Infrastructure Systems

by

Justin Burns, Thanh Bui, and ***Brajendra Panda***

Department of Computer Science and Computer Engineering
University of Arkansas

USA

SECUREWARE 2021, The Fifteenth International Conference on Emerging Security Information, Systems and Technologies

Agenda

- Introduction
- Related work
- Objectives
- Model
  - ✓ Graph and Metrics Definitions
  - ✓ Model Construction
  - ✓ System Repair Algorithm
- Conclusion

Introduction

- Critical infrastructure systems are those that are considered extremely critical to the function of a government or a country.

- Many of these systems are now connected to the internet, which makes them vulnerable to attacks.

- Prevention methods are not enough to fully secure a system and it is necessary to prepare for post attack activities, including damage assessment and recovery.

Related Work

- **Evaluating Synergistic Effects of Failures in CIS** [*Rehak et al.*, 2016]:

  Establishes a model with elements and linkages of varying criticality.

- **Post-Failure Network Recovery** [*Bartolini et al., 2016*]:

  Efficiently restores damaged element paths by recursively breaking demand flows into smaller problems.

- **Recovery with Progressive Damage Assessment** [*Ciavarella et al.*, 2017]:

  Uses centrality rankings to determine which damaged elements should be repaired first.

Objectives

- Establish a model that allows for fast, accurate, and efficient damage assessment.
- Define methods to give repair priority to more important sections of a system based on criticality and centrality metrics.
- Develop an algorithm to determine repair order using the aforementioned methods.

Model Definitions: Graphs

- Given two objects O_i and O_j in a system, if the value of O_j is calculated using the value of O_i, we say that there is information flow from O_i to O_j.
  - This defines the concept of information flow in a system.
- The **Possible Paths Graph (PPG)**
  - For a node N in each object O, there exists an edge E_{ij} between N_i and N_j if there is a possibility that N_j can be modified by N_i

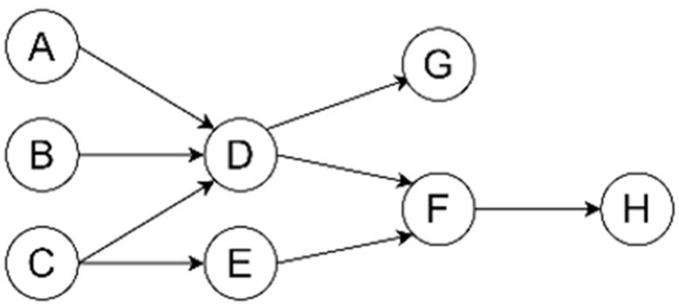

Figure 1a. The Possible Paths Graph (PPG)

Model Definitions: Graphs

- The **Active Paths Graph (APG)**
  - Each edge E_{ij} in the APG represents actual information flow between N_i and N_j. The APG must fully exist within the PPG
- The **Damage Spread Graph (DSG)**
  - Contains edges and nodes that have been damaged through information flow. The DSG must fully exist within the APG

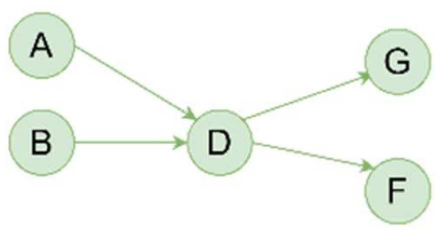

Figure 1b. The Active Paths Graph (APG)

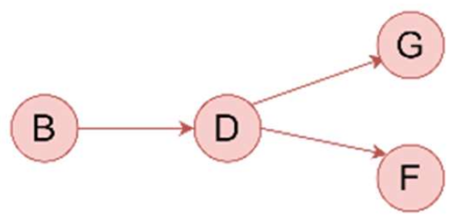

Figure 1c: The Damage Spread Graph (DSG)

Model Definitions: Metrics

- The **criticality** of a node *N* is its predetermined level of importance to the system's functions.
- The **repair time** of a node *N* is how many inward-flowing edges E^i it is receiving damage from.
  - For a node to be repaired, all of its parent nodes must be repaired first.
- The **centrality** of a node *N* is the number of outward-flowing edges E^o it has.
  - Nodes with higher centrality will reduce the repair time of more nodes when repaired than nodes with lower centrality.

Model Description

- We use the three defined graphs to model damage assessment and prepare for recovery
  - The PPG is preprocessed with all nodes and dependency paths in a system.
  - The APG is built by including the all paths that were used in the period between the initial attack and the current time.
  - Finally, the DSG is built by following the dependency paths that use the initially damaged nodes to make updates. Damage spread must follow two criteria:
    - There is a damaged node N_i that has an edge E_{ij} flowing from it to node N_j
    - E_{ij} is used for a transaction while N_i is damaged

Model Description

- ➢The goal of the model is to find the optimal sequence of repairs to restore the most important operations of a system as quickly as possible.
- ➢We must first plan to repair nodes in order of their criticality.
- ➢Between two or more nodes with equal criticality, the one with the lowest repair time is selected.
- ➢If two or more nodes also have an equal repair time, the one with the highest criticality is selected.
  - When the first node is picked, its parent nodes must be fully repaired first before it can be repaired.
  - This selection process repeats for all the parent nodes of the first node until a repair can be made.

Notations Table

Notations	Descriptions
$P = (V, E)$	Possible Path Graph
$A = (V_A, E_A)$	Active Path Graph ($V_A \subseteq V, E_A \subseteq E$)
$D = (V_D, E_D)$	Damage Spread Graph ($V_D \subseteq V_A, E_D \subseteq E_A$)
$D = (V_C, E_C)$	Critical Node Graph ($V_C \subseteq V_D, E_C \subseteq E_D$)
δ_{ij}	Decision to fix edge i to j
δ_i	Decision to fix node i
t_i	Time to fix node i
c_i	Centrality of node i
P_{ij}	Dependency indicator of node i and j

Our objective is to find min $\sum_{i \in V_D} t_i \delta_i$ subject to:

- $\delta_i \sum_{j \in V_C} P_{ij} \leq \sum_{j \in V_C} P_{ij} \delta_j \quad \forall i, j \in V_C$ (1)
- $\delta_i c_i \geq \sum_{(i,j) \in E_C} \delta_{ij} \quad \forall i \in V_C$ (2)
- $P_{ij} \in \{0,1\} \quad \forall i, j \in V_C$ (3)
- $\delta_i, \delta_{ij} \in \{0,1\} \quad \forall i \in V_C, (i,j) \in E_C$ (4)

Initialization Algorithm

Algorithm 1: Initialization for object set repair
Result: Queue of objects ordered by repair priority
1 Initialize set of damaged objects *O*
2 Preprocess object priority using criticality, repair time, and centrality
3 Initialize repair queue *Q*
4 while *O has damaged nodes remaining*
  4.1 Select the highest critical node(s) *N* within *O*
  4.2 if *Two or more nodes are tied for highest criticality*
    4.2.1 Select the node(s) *N* with the lowest repair time *R* within *O*
  4.3 if *Two or more nodes are tied for lowest repair time*
    4.3.1 Select the node(s) *N* with the highest centrality within *O*
  4.4 if *Two or mode nodes are tied for highest centrality*
    4.4.1 Select a single node at random from those still tied
  4.5 Update repair queue(N_0, O, Q) $\rightarrow$ Q
5 Print *Q*

Recursive Repair Algorithm

Algorithm 1.1: Recursive repair function
Result: Schedules a node N for repairs and returns the updated repair queue Q
1 Update repair queue(*Selected node N, object set O, repair queue Q*)**:**
2 while *Current object has unrepaired dependencies*:
  2.1 Create subset of damaged nodes O' of all nodes N' and edges E' that N is dependent on
  2.2 Select the highest critical node(s) N' within O'
  2.3 if *Two or more nodes are tied for highest criticality*
    2.3.1 Select the node(s) N' with the lowest repair time R within O
  2.4 if *Two or more nodes are tied for lowest repair time*
    2.4.1 Select the node(s) N' with the highest centrality within O
  2.5 if *Two or mode nodes are tied for highest centrality*
    2.5.1 Select a single node at random from those still tied
  2.6 Update repair queue(N'_0, O', Q) → Q
  2.7 Remove the most recent object in repair queue from O
3 Repair N
4 Add N to Q
5 Return Q

Conclusion

- We have presented a method to repair data objects that prioritizes quick recovery for the most important components of a system.
- This allows for the partial restoration of functions during the recovery process with an emphasis on restoring service to the most necessary functions.
- Our work is most applicable to protecting critical infrastructure systems where services need to be restored as quickly as possible to avoid economic or societal disruptions.