Technical Disclosure Commons

Defensive Publications Series

October 03, 2019

Predicting The Impact Of Maintenance Work On Networks

N/A

Predicting The Impact Of Maintenance Work On Networks

ABSTRACT

In large or complex distributed systems and networks, it is difficult to predict the impact of a maintenance task. Sometimes, multiple maintenance tasks are carried out concurrently, increasing the chance of outage due to inadvertent interaction between the maintenance tasks. In a bid to minimize the chance of outage, network managers often approach network maintenance in a conservative manner, resulting in a loss in the velocity of maintenance changes. This disclosure describes techniques that automatically vet the impact of network change requests by predicting network topology, modeling demands, simulating the effect of the change request, and reporting the impact back to the change management system for approval of the change.

KEYWORDS

● Network management
● Change request (CR)
● Change impact assessment
● CR velocity
● Network outage
● Network simulation
● Software-defined network (SDN)

BACKGROUND

In large or complex distributed systems and networks, it is difficult to predict the impact of a maintenance task. Sometimes, multiple maintenance tasks are carried out concurrently, increasing the chance of outage due to inadvertent interaction between the maintenance tasks. At large scale, predictions based on human experience or insight don't work, and can become a bottleneck to the velocity of change. In a bid to minimize the chance of outage, network managers often approach network maintenance in a conservative manner, resulting in a loss in the velocity of maintenance changes. In contrast, a network manager who seeks to satisfy the demand for network change requests at the speed of their arrival faces an increased risk of outages.

DESCRIPTION

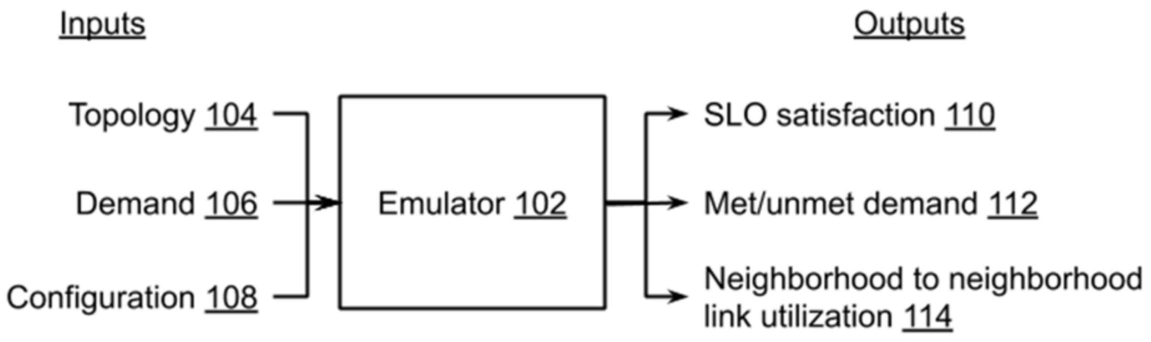

Fig. 1: An emulator that predicts the impact of maintenance works on networks

As illustrated in Fig. 1, per the techniques of this disclosure, a network emulator (102) mirrors the actual network being maintained. The actual network can be, for example, a software-defined network (SDN), a backbone network, etc. Upon receiving a network production change request (PCR), the PCR is automatically simulated on the network emulator using the following as inputs:

- Active, e.g., presently configured, and intended network topology (104) including the PCR delta and any concurrent PCRs.
- End-to-end traffic demand models (106), trended forecast, historical usage, current usage, etc.

- Network configuration (108), including bandwidth enforcement settings, traffic engineering settings, etc.

Example PCRs include the insertion or removal of an optical fiber link, insertion or removal of a router or switch, deployment of new capacity, etc. PCRs are made on a hot network, e.g., without shutting down, even temporarily, the network or parts thereof.

The output of the emulator includes the following.

- The fraction of service-level objectives (SLO) predicted to go unfulfilled due to the change request (110). SLOs include protected and unprotected bandwidths; the impact on each of these is separately reported.
- The fraction of unmet usage or demand (112).
- The location of the impact of the change request, including the neighborhood-to-neighborhood link utilization (114).
- Ancillary data, e.g., the network partition; data related to the simulation, e.g., times of start and end of the simulation; etc.

The simulation is run continuously while the change is applied, using the most up-to-date information available in the change management system. By doing so, unplanned changes, e.g., links going down in the real network without warning, are effectively incorporated into the simulation. The simulation can be run multiple times, including shortly before the PCR is actually implemented, such that the network change impact is assessed under the latest network conditions.

If the impact during a change management window is observed to be too large, the change is automatically halted and a notification is sent to the responsible party, e.g., the executor of the changes. If the impact is assessed by the responsible party to be too great, e.g., a drop in protected bandwidth is greater than 1%, then the responsible party can request that the change be done at another time of day, or be done in parts. The new request is again automatically simulated by the emulator and reported upon.

The emulator, which uses the same networking stack as the real (production) SDN stack and is integrated with the change management system, is continuously tuned and validated by loading a snapshot of network inputs and comparing the results of the simulation with real-life ones. Similarly, an emulator prediction of network behavior that does not pan out on the real network triggers an analysis of the causes of the apparent emulator inaccuracy.

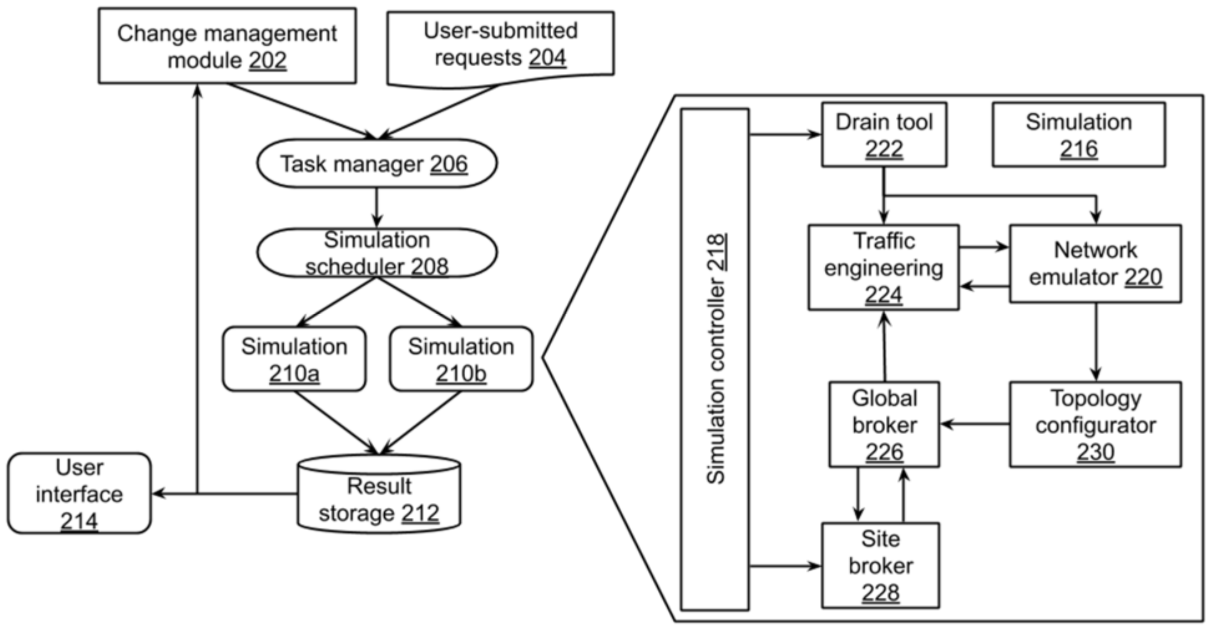

Fig. 2: Workflow to automatically assess the impact of change requests on a network

Fig. 2 illustrates an example workflow to automatically assess the impact of change requests on a network. A change management module (202) manages change requests. The change management module can make changes to the real network; however, per the techniques herein, change requests submitted to the change management module are vetted via simulation prior to getting approved. A network manager can also manually turn in change requests (204) to the emulator. This is useful to study network behavior in hypothetical scenarios. A task manager (206) queues tasks based on the resources available to the emulator. The task manager creates a simulation scheduler (208) that schedules simulations (210a-b) based on certain criteria, e.g., optimization of total (or mean or maximum) time taken by concurrent simulations, relative priorities of simulations, etc.

A simulation (216) is spawned. Within a simulation, a simulation controller (218) coordinates the various modules, e.g., a drain tool (222) to apply the PCR, a traffic engineering module (224) to provide traffic models, a global broker (226), a site broker (228), topology configurator (230), etc. that interact with the network emulator (220) to execute the simulation. The simulation is automatically run with and without the PCR, such that the impact of the PCR is brought out in the simulation output.

The output of the simulation is sent for results storage (212), from where it can be viewed via a user interface (214). The output of the simulation is also sent to the change management module, which can use the result to automatically approve (with or without modification) or deny the change request. The decision of the change management module is communicated to the responsible party, e.g., the executor of the changes.

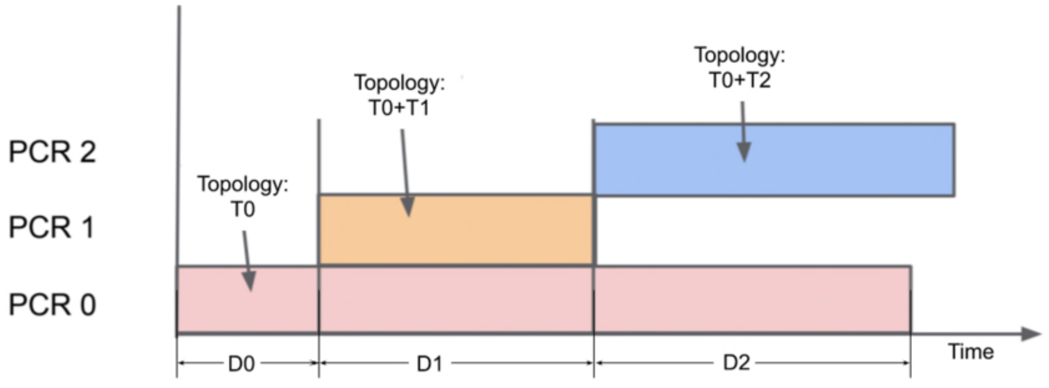

Fig. 3: Segmented PCRs

Among the features of the simulation framework described herein is the ability to handle concurrent, segmented PCRs, as illustrated in Fig. 3. In this scenario, PCRs last for limited durations. For example, PCR-0 applies for durations D0, D1, and D2. PCR-1 applies only for duration D1. PCR-2 applies only for duration D2. Thus, if PCR-0, PCR-1, and PCR-2 respectively induce topologies T0, T1, and T2 on the network, then for duration D0, the network topology is T0; for duration D1, the network topology is T0+T1; and for duration D2, the network topology is T0+T2.

Per the disclosed techniques, the emulator adjusts its topology in accordance with the applicable duration of concurrent PCRs. For example, in duration D0, the emulator uses the network topology T0; in duration D1, the emulator uses the network topology T0+T1; and in duration D2, the emulator uses the network topology T0+T2. By doing so, an accurate picture emerges of network behavior, compared to, for example, assuming a network topology T0+T1+T2 for all time, an over-conservative model that results in low velocity of change with no reduction in risk of outage.

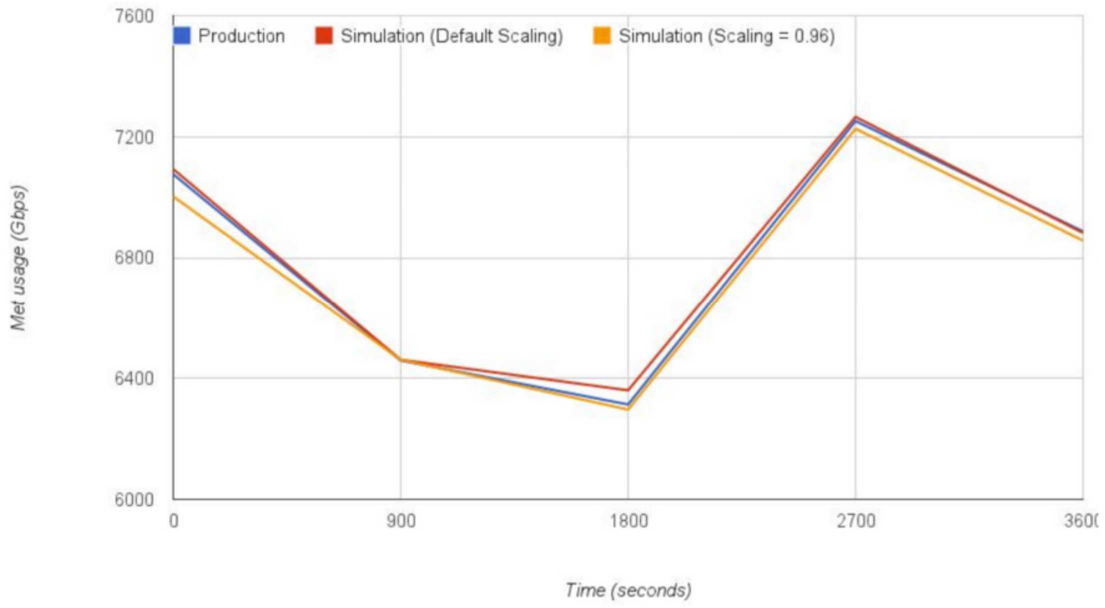

Fig. 4: A comparison between real and emulated network performance

Fig. 4 illustrates an example comparison between real (blue) and emulated network performance, where the parameter being compared is the met demand (usage) against time. The emulated network is run under two scaling parameters (red and orange), and in both cases, the emulated network performance is seen to be close to the performance of the real network.

In this manner, the techniques described herein can help achieve a balance between the velocity of the change requests and the safety of the changes based on a fast, accurate software simulation of the real network that produces a deterministic prediction of the impact of the change. The techniques eliminate potential outages and increase the number of overlapping changes, thereby improving network availability, network reliability, and velocity of change requests while decreasing human operator involvement.

CONCLUSION

This disclosure describes techniques that automatically vet the impact of network change requests by predicting network topology, modeling demands, simulating the effect of the change request, and reporting the impact back to the change management system for approval of the change.