Pod network loss in chaos engineering refers to intentionally inducing network packet loss within a Kubernetes cluster to simulate network-related faults and evaluate the system's resilience. This technique helps assess how well the system handles scenarios where network packets are dropped or lost due to network congestion, misconfigurations, or other issues.

Here's how pod network loss can be implemented in chaos engineering experiments:

1. \*\*Packet Dropping\*\*: Introduce faults in the network infrastructure or network overlay used in the Kubernetes cluster to drop network packets intentionally. This can be achieved through network proxies, traffic shaping tools, or by modifying network configurations to discard packets.

2. \*\*Traffic Injection\*\*: Inject packet loss at specific points in the network topology, such as between pods, nodes, or clusters. This simulates scenarios where network packets are dropped due to network congestion, hardware failures, or transient network issues.

3. \*\*Randomization\*\*: Randomly drop network packets or based on predefined criteria during chaos engineering experiments to simulate realistic scenarios. This helps uncover potential issues related to packet loss that may arise under varying conditions and workload patterns.

4. \*\*Observation and Analysis\*\*: Monitor the behavior of applications, services, and the underlying infrastructure during the network loss experiment. Collect metrics related to packet loss rates, latency, throughput, and application performance to assess the impact of network packet loss on the system.

5. \*\*Validation\*\*: Validate the system's resilience and fault tolerance by analyzing how well it handles network packet loss. Evaluate whether applications gracefully handle packet loss, recover from failures, or fail over to alternative resources, and identify any areas for improvement in the network infrastructure or application design.

By intentionally inducing pod network loss in chaos engineering experiments, organizations can proactively identify and mitigate potential issues related to network reliability, packet loss, and network congestion in Kubernetes clusters, thereby improving the overall resilience and robustness of their systems.

|  |  |  |
| --- | --- | --- |
| **Variables** | **Description** | **Notes** |
| NETWORK\_INTERFACE | Name of ethernet interface considered for shaping traffic |  |
| TARGET\_CONTAINER | Name of container which is subjected to network loss | Optional | Applicable for containerd & CRI-O runtime only. Even with these runtimes, if the value is not provided, it injects chaos on the first container of the pod |
| NETWORK\_PACKET\_LOSS\_PERCENTAGE | The packet loss in percentage | Optional | Default to 100 percentage |
| CONTAINER\_RUNTIME | container runtime interface for the cluster | Defaults to containerd, supported values: docker, containerd and crio for litmus and only docker for pumba LIB |  |
| SOCKET\_PATH | Path of the containerd/crio/docker socket file | Defaults to /run/containerd/containerd.sock |  |
| TOTAL\_CHAOS\_DURATION | The time duration for chaos insertion (seconds) | Default (60s) |  |
| TARGET\_PODS | Comma separated list of application pod name subjected to pod network corruption chaos | If not provided, it will select target pods randomly based on provided appLabels |  |
| DESTINATION\_IPS | IP addresses of the services or pods or the CIDR blocks(range of IPs), the accessibility to which is impacted | comma separated IP(S) or CIDR(S) can be provided. if not provided, it will induce network chaos for all ips/destinations |  |
| DESTINATION\_HOSTS | DNS Names/FQDN names of the services, the accessibility to which, is impacted | if not provided, it will induce network chaos for all ips/destinations or DESTINATION\_IPS if already defined |  |
| SOURCE\_PORTS | ports of the target application, the accessibility to which is impacted | comma separated port(s) can be provided. If not provided, it will induce network chaos for all ports |  |
| DESTINATION\_PORTS | ports of the destination services or pods or the CIDR blocks(range of IPs), the accessibility to which is impacted | comma separated port(s) can be provided. If not provided, it will induce network chaos for all ports |  |
| PODS\_AFFECTED\_PERC | The Percentage of total pods to target | Defaults to 0 (corresponds to 1 replica), provide numeric value only |  |
| LIB | The chaos lib used to inject the chaos | Default value: litmus, supported values: pumba and litmus |  |
| TC\_IMAGE | Image used for traffic control in linux | default value is gaiadocker/iproute2 |  |
| LIB\_IMAGE | Image used to run the netem command | Defaults to litmuschaos/go-runner:latest |  |
| RAMP\_TIME | Period to wait before and after injection of chaos in sec |  |  |
| SEQUENCE | It defines sequence of chaos execution for multiple target pods | Default value: parallel. Supported: serial, parallel |  |