Failure-Resilient Routing for Server-Centric Data Center Networks with Random Topologies

Ye Yu
Department of Computer Science
University of Kentucky
Lexington, KY, USA
ye.yu@uky.edu

Chen Qian
Department of Computer Engineering
University of California Santa Cruz
Santa Cruz, CA, USA
cqian12@ucsc.edu

***Abstract*—Data center networks with random topologies provide several promising features, such as near-optimal bisection bandwidth and flexibility of incremental growth. However, network failures are ubiquitous and significantly affect the performance of cloud services, such as availability and bandwidth. Existing random topology based data centers do not provide specific fault-tolerance mechanisms to recover the network from failures and to protect network performance from downgrading. In this work, we design FTDC, a fault-tolerant network and its routing protocols. FTDC is developed to provide high-bandwidth and flexibility to data center applications and achieve fault tolerance in a self-fixing manner. Upon failures, the nodes automatically explore valid paths to deliver packets to the destination by exchanging control packets. FTDC is a generalized solution that can be applied to multiple existing data center topologies. Experimental results show that FTDC demonstrates high performance with very little extra overhead during network failures.**

I. Introduction

Data center networks, as important communication and computing infrastructures, are playing unique and increasingly important roles in supporting cloud services and big data applications. These applications require data centers provide high-performance and reliable service.

Recent studies [14, 15, 16, 19] show that data center networks with random connections provide a number of promising features compared to traditional multi-rooted tree topologies such as FatTree [2, 3]. Random topology data centers achieve higher bisection bandwidth, higher throughput, and shorter routing paths [14, 15, 16, 19]. Meanwhile, random topologies support incremental growth of network size, i.e., adding servers or switches to the network without destroying the current topology or replacing the switches, while multi-rooted trees cannot do so due to their “firm” structures. In addition, random topologies can be used for either switch-centric networks, in which servers can interconnected by high-capacity switches, or server-centric networks, in which servers are connected directly [1, 7, 8, 14]. Typical random topologies are Small-World Data Center (SWDC) [14], Jellyfish [16], and Space Shuffle (S2) [19]. They all supports both types of networks.

Similar to any other network infrastructure, data center suffers from network failures [6]. In data centers, both switches and servers in the network may fail independently [5]. A typical new data center may encounter hundreds of switch failures and thousands of individual server failure each year [6]. Some typical types of failures are hard drive failures, memory failures, and flaky machines. Link failures are also common in data center networks [6], which are unpredictable and usually burst unexpectedly.

Failures are harmful to cloud services. Upon failure, if not handled correctly and effectively, the performance of the cloud may be severely impacted. First, the connectivity of the data center topology may be damaged due to switch, link, and server failures. This may result in loss of routing correctness and availability of the cloud service. Second, such failure may lead to rerouting or congestion in the network and hence network bandwidth is likely to be lost. Third, the failure may also cause the imbalance of load among nodes. Hence, it is critical to improve the robustness of cloud data center networks to fit the reliability requirements of cloud services. We summarize the robustness requirements of a data center network as follow:

- Network failures, such as link failure, should be discovered timely so that the network administrator can be informed in time and take actions to fix the failure.
- Upon a failure, an alternative routing mechanism should be applied, and the network should still guarantee that the packets are delivered correctly.
- The failure should not severely impact the network throughput and latency.

Note that this work only discusses the packet delivery in *the network layer*. That is, upon a failure, the remaining network devices can still route a packet to the destination using their local information in a distributed manner. Reliable packet delivery in other layers such as TCP is out of the scope.

Some studies aim at providing failure tolerance to data centers. For hierarchical topology data center built with of a large number of switches, the follow-up project of Fat-Tree, Portland [12] provides a fault-tolerant Layer 2 routing

9781-5090-1445-3/16$31.00 © 2016 IEEE (CloudCom'16)

protocol. F10 [11] further improved the fault tolerant capability. However, there is no comprehensive study that aims on improving failure resiliency for random topology data centers. In fact, this is more challenging to find alternative paths in these networks due to their random nature. Some studies claim certain levels of fault-tolerance for these data centers[14, 16, 19]. However, most of them only guarantee that there exists a path from a source to any destination upon failures. None of them satisfies all above requirements.

In this paper, we design a fault-tolerant data center network architecture including failure-resilient routing protocols, called FTDC. FTDC is the first work to provide *high-bandwidth and flexibility* to data centers with *random topologies* while achieving the above three fault-tolerance requirements. Upon failures, the forwarding nodes do not wait until human-efforts are performed to fix the routes. They automatically explore valid paths to deliver packets to the destination by exchanging control messages among servers. We take the recent random-connected data center design Space Shuffle (S2) [19] as the underlying topology. Upon failures, FTDC tries to find an alternative path for a flow by deploying a revised greedy routing on particular packets. The rest packets in the flow are routed along the alternative path. FTDC provides short alternative path effectively. It only requires trivial modification on the packets, which can be implemented using the VLAN tag of the Ethernet frame.

As greedy routing are more common in server-centric data centers than switch-centric ones, in this paper, we slightly modify the switch-centric S2 topology as a server-centric network, and apply the fault-tolerant designs. This method can also be *directly applied to either server-centric or switch-centric network that adopts random topology and greedy routing*. We call a forwarding device "node", which refers to either a switch or a server.

The rest of the paper is organized as follows. We describe the background and our motivation in Section II. We present the FTDC design in Section III. We describe the routing protocol in Section IV. We evaluate the performance in Section V. We discuss a number of related works in Section VI and finally conclude this work in Section VII.

II. Background and Motivation

A. Data Centers with Random Topologies

Recently, researchers have found that random links in data center help to reduce routing latency and they increase bisection bandwidth and throughput. Moreover, random connections provide the flexibility of incremental growth of network size [16, 19]. The Small-World Data Centers (SWDC) [14] are built with directly connected servers in three types: ring, 2D Torus, and 3D Hex Torus, as well as random connections to improve network performance. SWDC uses geographic routing with Euclidean distance. Jellyfish [16] uses completely random connected networks to achieve higher network throughput compared to previous designs. However, it has no explicit routing protocols and relies on the centralized computation of k-shortest path, which is inefficient in both time and forwarding state size. Space Shuffle (S2) [19] uses random connections in multiple ring spaces. It adopts geographic routing in ring spaces.

Switch-center versus server-centric. Sever-centric data centers demonstrate several advanced features in multiple data center applications, including network-layer key-value query [1] and cost-efficiency due to the removal of expensive high performance switches. SWDC, Jellyfish, and S2 [14, 16, 19] can all be used for server-centric networks, where a server that performs forwarding can be considered a switch with a small number of ports. In random networks, all forwarding nodes perform a same role without heterogeneity.

Fault-tolerant data center networks. There have been several types of research on fault-tolerant data center networks. DCell [7] executes local rerouting when a failure is detected in the network. However, it does not guarantee loop freedom under multiple failures. F10 [11] re-designed the multi-root tree topology, which provides interlaced alternative paths. F10 designed a set of protocol to ensure fast failover. Generally, greedy routing protocols do not guarantee packet delivery unless with the help of the topology properties. MDT [10] executes greedy routing on a Delaunay Triangulation of the coordinate space and guarantees packet delivery. On the other hand, Data-Driven Connectivity design (DDC) [18] aims to provide connectivity with data plane mechanisms. However, none of them meets the four requirements of a fault-tolerant data center network with random connections.

B. Geographic Greedy Routing

Geographic routing is a routing principle in which the routing decision is made according to the position information (i.e., coordinates) of the destination and the neighboring nodes. It has been widely adopted in sensor or mobile ad-hoc networks [17] and some other large networks [13]. The routable address of a node in the network can be derived from its physical location [9] or virtual coordinates [4]. Using geographic routing protocols, each node only stores information about its adjacent neighbors, other than a forwarding table covering network-wide addresses. Hence, the data plane storage overhead of greedy routing is significantly small. Geographic routing is a scalable and practical routing solution for random topologies [19].

Choosing a proper metric to compute the distance between nodes is critical in greedy routing. Suppose a node X uses metric function $\mathcal{M}$ as the distance metric, when X receives a packet with destination coordinate D, it computes the distance between the candidate neighbors and the destination for all candidates $Y_1, Y_2, \cdots \in C_X$ (C_X is the candidate set of X). Then the packet is forwarded to the one neighbor $Y_{\mathcal{M}}$, $Y_{\mathcal{M}}$ has the minimal distance to the destination based on metric $\mathcal{M}$. i.e.,

$$\mathcal{M}(\overrightarrow{Y_{\mathcal{M}}},D)=\min_{Y_i\in C_X}\mathcal{M}(\overrightarrow{Y_i},D) \tag{1}$$

Note that, the next-hop neighbor Y_n should be closer to the destination than the current node X. i.e. , greedy routing requires that

$$\mathcal{M}(\overrightarrow{X},D)>\mathcal{M}(\overrightarrow{Y_{\mathcal{M}}},D)$$

If X is closer to the destination than all the candidates, i.e. $\mathcal{M}(\overrightarrow{X},D)\le\mathcal{M}(\overrightarrow{Y_{\mathcal{M}}},D)$, the packet may be forwarded back to X by Y_n, because X is closer to the destination than X. We say that the current node X is a *local minimum* of the distance metric $\mathcal{M}$ to destination D.

C. Space Shuffle (S2) Data Center Network

In this paper, we use the Space Shuffle (S2) topology [19] as the underlying topology for FTDC. S2 adopts a multi-ring random switch-centric topology. Each switch is connected to $2L$ neighbor switches, and the rest of the ports are used to connect single-homed hosts. Each switch is assigned with a L-dimensional vector $\langle x_1,x_2,\cdots,x_L\rangle$, $(0\le x_i<1\ ,1\le i<L)$. Each x_i represents the position of the switch on the i-th ring and the switches are connected accordly. Two switches are connected if they have adjacent coordinates in at least one space. The switches form L rings.

S2 adopts greedy geographical routing and uses Minimum Circular Distance (MCD) as the distance metric. *MCD* is defined as follows. For any two between two coordinate $\overrightarrow{X}=\langle x_1,x_2,\cdots,x_L\rangle$ and $\overrightarrow{Y}=\langle y_1,y_2,\cdots,y_L\rangle$, the MCD between X and Y is

$$MCD(\overrightarrow{X},\overrightarrow{Y})=\min_{1\le i\le L}CD(x_i,y_i)$$

where $CD(x,y)=\min\{|x-y|,1-|x-y|\}$.

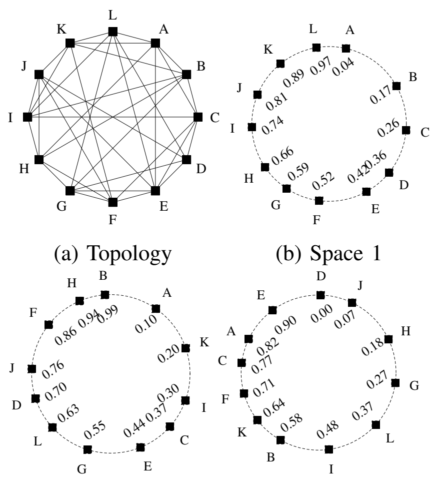
III. Design Overview

An FTDC is an interconnection of nodes. Each node is a server equipped with multiple NICs to connect to its neighbors (server-centric), or a Top of Rack (ToR) switch (switch-centric). FTDC adopts greedy geographical routing. All nodes execute the identical routing protocol.On receiving a packet, the node identifies whether itself is the destination of the packet, and it forwards the packet to the next-hop node, or accepts the packet. The robust routing protocol of FTDC extends the geographical routing protocol by supporting additional shortcut paths upon network failure.

A. FTDC Design Goal

We categorize network failures of a data center into two classes: (1) *link failure*: a link between a pair of nodes fails. A node can not communicate with other nodes through that particular link. (2) *node failure*: During a node failure, a node does not send, receive, or forward packets correctly.

FTDC maintains the network functionality in the presence of link failures and node failures. The nodes in FTDC ping each other periodically to detect the failures. It is also assumed that the network failures happens independently so that failures in a small portion of the network does not cascade to other parts of the network.



(a) Topology (b) Space 1

(c) Space 2 (d) Space 3

Figure 1: An Example FTDC Topology. (a): Wire connection among nodes. (b),(c),(d): Coordinates in different spaces.

The goal of FTDC is to build a robust network infrastructure that handles network failure efficiently and maximizes packet delivery. The FTDC routing protocol works collaboratively with the existing greedy geographical routing protocol. FTDC is *transparent* to the data center applications running on end hosts. FTDC *guarantees to establish a shortcut path* to the destination of a packet effectively as long as the destination is still reachable.

B. FTDC Network Topology and Greedy Routing Protocol

Similar to S2, each node in FTDC is connected to $2L$ neighbors directly. L is a parameter of the topology, identifying the number of virtual spaces. For a typical server-centric data center, each node is equipped NICs with 6 ports and $L=3$. Each node is assigned with an L-dimensional coordinates $\langle x_1,x_2,\cdots,x_L\rangle$, where $0\le x_i<1$ $(1\le i\le L)$. Fig. 1a shows an example of an FTDC topology of 12 nodes. Most of the links in Fig. 1a connect two nodes that are adjacent in at least one *virtual space* in Fig. 1b, 1c or 1d. We call them *link on circles*. Other links are randomly selected to vitalize the free ports of the switches, they connect nodes A/H, B/G, D/G, E/J and E/L.

FTDC uses the *minimum circular distance* (MCD) as the routing metric of greedy geometry routing. On receiving a packet, a node first checks whether itself is the destination of the packet. If the destination of the packet is not the current node , the node will execute the greedy forwarding procedure: it computes the MCD between the candidate neighbors and the destination of the packet and then select the neighbor with minimal MCD to the destination as the next hop node.

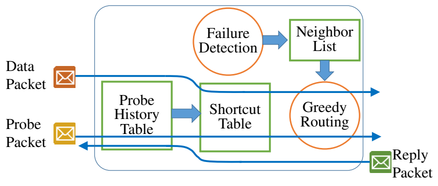


Figure 2: FTDC Data Plane Pipeline

When a node computes the next hop for a particular packet, all of the neighbors of the current node are considered to be valid candidates. The property the topology guarantees that if a node is not the destination of a packet, one of its neighbors must have a smaller *MCD* to the destination than the node. Therefore, greedy routing using MCD as the distance metric guarantees packet delivery on the topology, as long as all the *links on circles* and all the nodes work normally [19].

In addition, the nodes are connected through each of the rings. Two nodes may be close to each other on one ring but far away on another. This fact suggests that we can find an alternative path by letting the packet traverse through one particular ring. FTDC utilizes this property of the topology to discover alternative paths during network failures.

C. *Overview of FTDC Failure-Resilient Routing*

An FTDC node detects failures by sending echo packets to its neighbors. The node maintains a list of active neighbors and executes greedy geographic routing. Upon failures, the integrity of the topology is damaged and is reflected in the active neighbor list of some nodes. Some nodes may become the local minimums of packets. These nodes are not the destination, but every neighbor node has a greater *MCD* to the destination than the current node.

The key principle of FTDC failure-resilient routing is to establish a shortcut for the packets that are impacted by a network failure *by carefully selecting the candidate next-hop nodes and the distance metric*. When a packet is routed to a local minimum node, the node will tag the packet as a probe packet and send replicas of the probe packet along particular alternative edges towards the destination.

Once a probe packet reaches the destination, a shortcut path is found. Then the destination server sends reply packets to confirm this shortcut. The shortcut path is used to route the following packets to the same destination. When the failure is fixed, FTDC recovers the entries in the active neighbor list and removes the related shortcut paths. The following packets are routed using the normal path.

IV. FTDC Failure-Resilient Routing Protocol

In this section, we explain the failure-resilient routing protocol of FTDC. It can be implemented in the protocol stack as a network layer protocol.

A. *FTDC Data Plane Pipeline*

Data packets in FTDC are routed by the nodes. Each node executes the same routing procedure. FTDC uses three packet types. A node forwards a packet according to the type and address information carried by the packet.

1) Packet Types:

- **Data Packets.** These are the packets sent and received by the applications. The destination coordinate is specified in the destination field of the packet. FTDC forwards these packets using greedy geographical routing, with MCD as the distance metric.
- **Probe Packets.** Probe Packets are used to discover a shortcut path for their following data packets. A probe packet is constructed by adding a tag to a data packet that identifies the failure-resilient routing mode should be used for this packet. The available options are $\mathtt{CD_k}$, $\mathtt{CD_k^+}$, $\mathtt{CD_k^-}$ and `BROADCAST`, where $1 \le k \le L$. A probe packet can be restored to a data packet by having the tag removed.
- **Reply Packets.** A Reply Packet is used to confirm a shortcut path. A reply packet carries the information of a pair $\langle \overrightarrow{D}, X \rangle$, which indicates that there is a path from X to the destination with coordinates $\overrightarrow{D}$.

2) Assisting tables: The failure-resilient routing scheme is a part of the routing protocol of FTDC nodes. Each node maintains three tables to assist the routing procedure.

- **Neighbor List.** A neighbor node is *available* means that this node and the link connecting the current node to it, are working normally as excepted. The set of available neighbors is the candidate set of greedy routing.
- **Shortcut Table.** This table is used to store the discovered shortcut paths for a destination. Each entry in this table is a pair $\langle \overrightarrow{D}, Q \rangle$. It identifies a shortcut path to the destination coordinate $\overrightarrow{D}$ from the current node via neighbor Q. Once a shortcut path is discovered, an entry will be added to this table so that the following data packets with the same destination can be forwarded along the shortcut path.
- **Probe History Table.** This table is used to eliminate redundant probe and reply packets. Each entry is a 3-tuple $\langle \overrightarrow{D}, \mathcal{M}, X \rangle$. It indicates three facts. (1) A probe packet with destination coordinate $\overrightarrow{D}$ and tag $\mathcal{M}$ has traversed through this node. (2) This probe packet is forwarded to the current node by node X. (3) The node has not received any reply packets from the destination D. All recurrence probing packets will be dropped by checking this table.

B. *Greedy Routing in FTDC*

Greedy routing in normal situations

In normal situations, FTDC executes the greedy routing method presented in Section III-B with MCD as the routing metric. We discuss the situation when failures exist in the network as follows. If no packets encounter local minimum, FTDC executes the same method as normal. Otherwise, FTDC uses the following schemes to send probe packets.

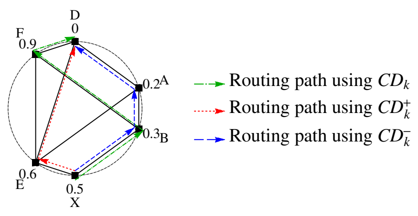

D
0
F
0.9
0.2A
0.3B
0.6
E
0.5
X
Routing path using CD_k
Routing path using CD_k^+
Routing path using CD_k^-

Figure 3: Examples of routing paths from X to D generated by CD_k, CD_k^+, and CD_k^-.

Alternative Routing Metrics. FTDC uses alternative distance metrics CD_k, CD_k^+ and CD_k^- to select next-hop nodes for probe packets. The distance metric CD_k is used to route packets in space k. CD_k^+ and CD_k^- are used to route the packet along the circle of space k in a given direction. For a node with coordinate $\overrightarrow{Y} = \langle y_1, y_2, \cdots, y_l \rangle$ and destination coordinate $\overrightarrow{D} = \langle d_1, d_2, \cdots, d_L \rangle$, they are defined as follows.

- $CD_k(\overrightarrow{Y}, \overrightarrow{D})$ is the circular distance measured on the k-th space between coordinate $\overrightarrow{Y}$ and $\overrightarrow{D}$.
- $CD_k^+(\overrightarrow{Y}, \overrightarrow{D})$ is measured clockwise from y_k to d_k.
- $CD_k^-(\overrightarrow{Y}, \overrightarrow{D})$ is measured counterclockwise from y_k to d_k.

$$CD_k(\overrightarrow{Y}, \overrightarrow{D}) = CD(y_k, d_k) = \min\{|y_k - d_k|, 1 - |y_k - d_k|\}$$

$$CD_k^+(\overrightarrow{Y}, \overrightarrow{D}) = \begin{cases} d_k - y_k & \text{if } y_k \le d_k \\ 1 - y_k + d_k & \text{if } y_k > d_k \end{cases}$$

$$CD_k^-(\overrightarrow{Y}, \overrightarrow{D}) = \begin{cases} 1 - d_k + y_k & \text{if } y_k < d_k \\ y_k - d_k & \text{if } y_k \ge d_k \end{cases}$$

Figure 3 shows an example of different routing paths.

(1) When CD_k is used, the packet is sent to the next-hop node that is closest to the destination measured in space k. Hence, we use CD_k to route a packet in space k.

(2) CD_k^+ and CD_k^- are used to restrict the direction of the packet. When CD_k^+ is used, only the neighbor in the clockwise direction of X may be possibly selected by X. The packet then traverses clockwise to the destination in space k. Similarly, when CD_k^- is used, the packet traverses counterclockwise in space k.

C. Probe for a shortcut path

When a packet with destination coordinate D is routed to its local minimum X in FTDC, node X initiates the probe process to establish a shortcut for the destination. Each of the probe packets is tagged with a field, identifying the distance metric that should be used when the rest nodes forward it.

The delivery property of MCD routing on FTDC network is affected by the network failure. Hence, the probe packets should detour along the network.

To utilize network redundancy across multiple spaces, we let the current local minimum node X send out multiple probe packets, one on each space. For each space k, if the circle in space k is integral (observed by X), we let the probe packet traverse along this circle using CD_k as the distance metric. If the circle is broken, we force the packet to traverse along a fixed direction to avoid the broken link on the circle, by using CD_k^+ or CD_k^- as the distance metric.

Each of the probe packets traverses along one particular space. These probe packets are expected to reach their destination D, or a midway node X which has an established shortcut to the destination. FTDC routes one or more replicas of the same data packet to the destination D. However, only the first replica is finally accepted by D while the others are dropped by the data plane of node D.

The correctness of routing using CD_k^+ or CD_k^- is presented as Lemma 1. The proof is omitted due to space limitation.

Lemma 1. *In an FTDC network, a link connects adjacent nodes X/Y in space i fails. X is a local minimum for greedy routing using MCD for a packet with destination D.*

If y_i is on the clockwise direction of x_i and all other links in space i still work normally, then greedy routing using CD_i^- routes the packet to the node with coordinate d_i in space i.

D. Shortcut path confirmation

Once a probe packet reaches the destination, the path that it traversed through is considered as the shortcut path of this destination. The destination replies a reply packet, which specifies the coordinate of the destination and the sequence number of the corresponding packet.

When a node receives a reply packet with destination coordinate D from its neighbor Y, it will add an entry in the shortcut table identifying the next-hop node for packets with coordinate D is Y. Future probe packets with coordinate D will be dropped since the shortcut is already established.

E. Concurrent Failures

The probe packets are routed using the metric CD_k, CD_k^+ or CD_k^-. Each of the probe packets carries a tag specifying which metric should be used to forward this packet. However, as a consequence of multiple coexistent failures, a probe packet may also be routed to its local minimum. In such circumstance, the current node can not select a next-hop node for the packet in space k. Instead, the current node sends probe packets through other spaces i ($i \neq k$), so that the probe process may continue. FTDC adopts a fail-safe mechanism to guarantee the establishment of a shortcut path as long as the destination is still reachable. When a node initiates a probe process for a shortcut, it sets up a timer and waits for the corresponding reply from the destination. Upon the timeout, the node sends another series of probe packets, namely the *broadcast probe packets*. Their tag fields are set to be `BROADCAST`. These packets traverse the network to probe for a possible shortcut. Each link in the network transmits at most one of these probe packets. The probe history tables on the nodes are used to prevent broadcast storm in the network.

F. Forwarding procedure

All the nodes execute the same forwarding procedure. On receiving each packet, the node executes the forwarding procedure according to the type of the packet. It decides whether to forward the packet to a neighbor node, or to accept the packet and hand it over to the application.

1) Data Packet Forwarding: When a node X receives a data packet r, The `Regular` procedure is executed. It tries to forward the packet using the Shortcut Table and MCD forwarding. If it fails to do so, it sends out replicas of r as probe packets to discover a shortcut.

In the case where X sends out probe packets, X waits for a certain amount of time *Timeout* and then checks whether a shortcut has been established. Here, *Timeout* is a parameter specified during node initialization. If the shortcut is still not available by then, X executes the fail-safe mechanism by sending out broadcast probe packets. The broadcast probe packets are sent to all of its available neighbors, other than the neighbor that r comes from. In the case when X has only one neighbor Y, one broadcast packet is sent to Y.

2) Probe Packet Processing: When a node X receives a probe packet, the `Probe` procedure is executed.

X first checks whether it is the destination, or there is an alternative shortcut path for $\overrightarrow{D}$ in the Shortcut Table. If so, X restores the probe packet r to its original data packet form. The packet r is then forwarded along the shortcut path or accepted by X. Then X sends a reply packet to confirm the establishment of a shortcut path to destination $\overrightarrow{D}$.

If r fails to pass the check, the node continues to check whether it has already seen a probe packet with identical tag and destination by looking up in the Probe History Table. X sends only the packets that have no identical precedents. Such action is reflected as an entry in the Probe History Table.

For a broadcast packet, X sends its replicas to all the available neighbors except the one which it comes from. For a probe packet r tagged with distance metric $\mathcal{M}$, X performs greedy routing using the metric $\mathcal{M}$. X sends probe packets for r if X is the local minimum of r. (Sec IV-E)

3) Reply Packet Processing: A reply packet always identifies a shortcut to a destination $\overrightarrow{D}$ via a node P. When a node X receives a reply packetit adds a new entry to the Shortcut Table, and a shortcut from the current node X to $\overrightarrow{D}$ is established. X also sends reply packets to the nodes who has previously sent probe packets to X for destination $\overrightarrow{D}$.To prevent redundant reply packets, the corresponding Probe History table entries are removed as well.

G. Estimating the next-hop coordinates

In an FTDC of n nodes with $L > 1$. Node X receives a packet with destination Y. The next-hop node W would satisfy $\mathcal{M}(\overrightarrow{X}, \overrightarrow{Y}) > \mathcal{M}(\overrightarrow{D}, \overrightarrow{Y})$. Here, we provide a quantitative estimation of $\mathcal{M}$, assuming that all the coordinates are uniformly indecently randomly assigned for all the node. Let $x = \mathtt{CD_k}(\overrightarrow{X}, \overrightarrow{Y})$. Without loss of generality, let $y_k = 0$.

When nodes X and Y are adjacent in space k, all the coordinates of other nodes do not lie in range $[0, x]$. Hence, node X and node Y is adjacent in space k with probability $(1-x)^{n-2}$. Otherwise, suppose the next-hop node on the path from X to Y is Z. Let $t = z_k$ be the coordinate of Z in space k. The cumulative distribution function of t is $F(t)$,

$$F(t) = (1-(x-t))^{n-2} \quad (0 \le t \le x)$$

The $\mathtt{CD_k}$ value between X and any one of the other $2L-2$ neighbors (denoted as W) satisfies distribution function $P(\mathtt{CD_k}(X, W) \le t) = 2t \ (0 \le t \le \frac{1}{2})$.

Let $\mathcal{N}$ be the next-hop node of X, and let E_{next} be the expected $\mathtt{CD_k}(\overrightarrow{\mathcal{N}}, \overrightarrow{Y})$. Note that E_{next} depends on x.

$$\begin{aligned} E_{\text{next}}(x) &= (1-(1-x)^{n-2}) \int_0^x t(1-(1-F(t)) \cdot (1-2t)^{2L-2})' \mathrm{d}t \\ &= \frac{1-(1-x)^{n-2}}{2} \Bigg(\frac{1-(1-2x)^{2L-2}(4(L-1)x+1)}{2L-1} \\ &\quad + x^2(1-x)^{n-2}\Big((L-1)(x-1)\mathcal{A}(2,3-2L,1-n,3) \\ &\quad + (n-1)(\frac{1}{4}\mathcal{A}(2,3-2L,2-n,3) - (\frac{x}{3}\mathcal{A}(3,3-2L,2-n,4)))\Big)\Bigg) \end{aligned}$$

Here, the notation $\mathcal{A}$ is the Appell series F_1. Since all the $\mathcal{A}$ items are negative in the above equation, we have,

$$E_{\text{next}}(x) < (1-(1-x)^{n-2}) \frac{1-(1-2x)^{2L-2}(1+4Lx-4x)}{2(2L-1)} < 2(L-1)x^2$$

Hence, the distance to the destination reduces quadratically.

H. Apply FTDC to Other Networks

FTDC can also be applied in other random-connection-based networks that use geographic routing. For example, SWDC [14] also provides multiple disjoint paths between two nodes, FTDC routing can automatically find a shortcut path when there is a node or link failure. For 1D-ring SWDC, identical protocol can be executed. For 2D or 3D, when there is a direction change in FTDC, the node should forward the packet via adjacent link on the other direction, by changing the distance metric accordingly. The topology of CamCube [1] can be considered a subgraph of SWDC, hence similar techniques can be applied.

V. Evaluation

We conduct extensive experiments to evaluate the efficiency and reliability of the FTDC design. We conduct fine-grained packet-level simulation using a lightweight simulator[1]. We simulate all packets in the FTDC network, including data packets, TCP control packets, probe packets and reply packets. Each of the plots in the figures is generated using at least 32 runs.

We use two models to simulate the failures. (1) **Random Link Failure**: A fraction of links in the network are marked as inactive and do not transmit packets. (2) **Random Node Failure**: A fraction of nodes do not send or receive packets.

We evaluate the following performance criteria of FTDC.

[1]We experienced very slow speed when using tools like NS for data center networks. We guess the existing studies [1, 14, 16] do not use NS due to the same reason.

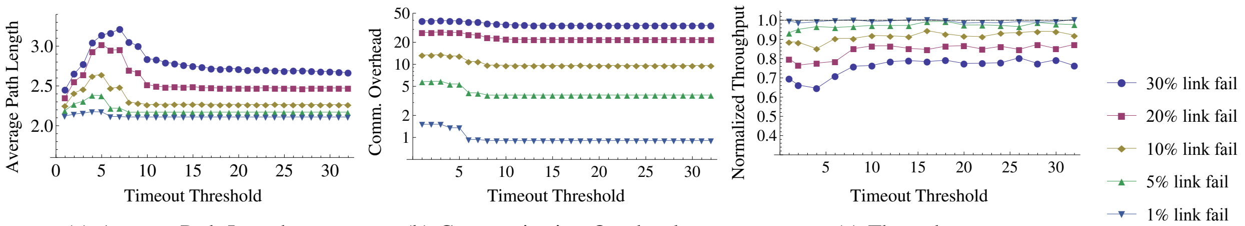

(a) Average Path Length (b) Communication Overhead (c) Throughput

Figure 4: Evaluation on Different Timeout Thresholds

- **Average Path Length**: We measure the lengths of all server-to-server paths. The average path length reflects the efficiency of the shortcut paths generated by FTDC.
- **Communication Overhead**: We measure the number of probe packets and reply packets transmitted on each link in the network for all-to-all traffic. The communication overhead is indicated by the average number of extra packets transmitted on each link.
- **Computation Overhead**: Compared to normal situations, a node makes extra effort when a packet is routed to a local minimum, or when it sends broadcast packets due to timeout. To reflect the computation overhead of FTDC failure-resilient routing, we measure how many times when function `StartProbe` is executed, and when a node sends out broadcast packets,
- **Storage Cost**: We measure the number of entries in the shortcut table on each node after all shortcut paths have been established for all-to-all traffic. This number reflects the cost to store the shortcut paths.
- **Throughput**: We measure the throughput of random permutation traffic on an FTDC network. It reflects how FTDC utilize the bandwidth of the topology.

Compared to other greedy routing networks: We conducted experiments to compare FTDC with two other data center networks that adopt greedy routing, namely S2 [19][2] and SWDC[14]. S2 and SWDC fail to deliver all the packets under network failure. Since the deliver rate of FTDC differs from S2 and SWDC, it is meaningless to compare their performance in other criteria such as the average forwarding path length or throughput.

A. *Evaluation on different Timeout thresholds*

We evaluate the FTDC performance under different *Timeout* settings. *Timeout* is a parameter that indicates how much time a node waits until it sends broadcast packets of a particular destination. (See Sec. IV-E)

Nodes are connected by links of the same bandwidth and latency in a 32-node network. They share the same *Timeout* value. We let this value vary and measure the path length, the communication overhead and the throughput of the network as shown in Fig. 4. We normalize the round-trip-time of a single link to 1 and use it as the unit of the x-axes.

[2]S2 is switch-centric. We construct a server-centric network using the same topology and routing protocols as S2 for evaluation.

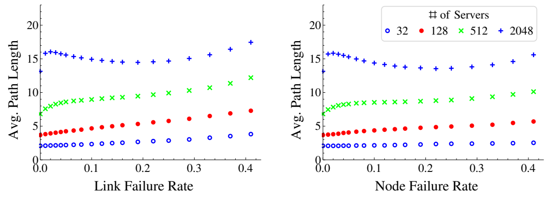


Figure 5: Average Path Length vs Failure Rate

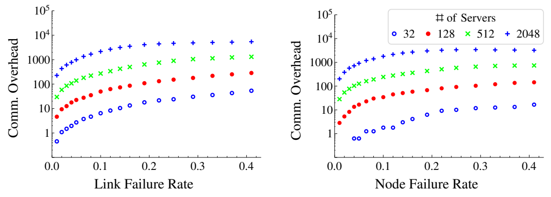


Figure 6: Communication Overhead vs Failure Rate

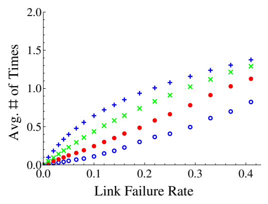
Larger *Timeout* values require smaller communication overhead (Fig. 4b) and provide better throughput (Fig. 4c) and average path length (Fig. 4a). Note that the latency of the other packets with the same origin/destination is not affected. Hence, we recommend larger *Timeout* values in practice.

B. *FTDC performance under failure*

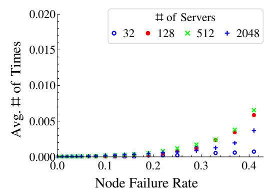
Average Path Length: Fig. 5 shows the average path length between all server-to-server peers. The path length grows slowly as more fraction of the network fails. Interestingly, when 10% to 30% of a 2048-node FTDC network fails, the path length is stable. This is because the FTDC topology provides abundant parallel paths, and these paths can be used as alternative shortcuts.

Communication Overhead: Fig. 6 shows the communication overhead of FTDC under all-to-all traffic. FTDC only introduces minor communication overhead into the network. Note that these probe and reply packets are only transmitted in the once upon failure occurs. FTDC does not require any additional communication once a shortcut is established. Even for a 2048-node network with 30% links fail, each link only transmits about two thousand packets.

Computation Overhead: We evaluate the computation overhead for FTDC under different failure rates in Fig. 7. Fig. 7a shows that `StartProbe` is executed for less than 1.5 times on average. Most of the probe packets



(a) Avg. times that `StartProbe` is executed



(b) Avg. times that broadcast probe pkts are sent

Figure 7: Computation Overhead vs Link Failure

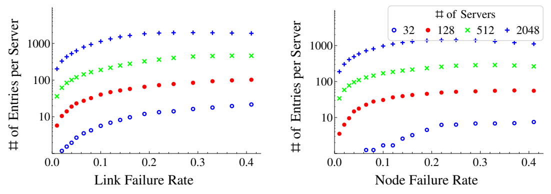


Figure 8: Storage vs Failure Rate

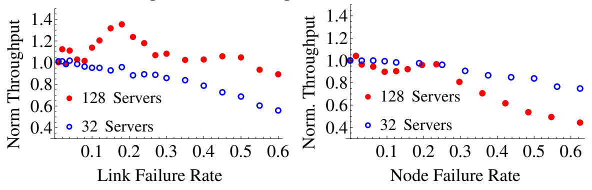


Figure 9: Throughput vs Fail Rate

reached the destination finally. The broadcast packets are only necessarily sent in a minor fraction of situations. Fig. 7b shows that the broadcast packets are sent out for about 0.005 times on average. This indicates about 99.5% of the probe packets reach the destinations without starting a broadcast.

Storage Cost: Fig. 8 shows the average number of entries in the Shortcut Table on each node. It grows linearly against the fraction of failure in the network. However, this overhead is still small and tolerable for memory.

Throughput: Fig. 9 shows the throughput of FTDC under random permutation traffics. The throughput of a network with no failure is normalized to 1. For a 32-node network with 30% links or nodes fail, FTDC still provides as much as 80% of the original bandwidth. Somewhat surprisingly, we found that the throughput for a 128-node network with failures is higher than the one with no failure. It is because the FTDC failure-resilient routing algorithm uses different methods to find paths under failure, which actually helps to reduce network congestion.

VI. Conclusion

The contribution of FTDC is to provide failure-resilient services in the network layer, for data center networks with random topologies that supports higher throughput and flexibility compared to multi-rooted trees. FTDC adopts greedy routing in normal situations and achieves fault tolerance in a self-fixing manner. Upon failures, the nodes automatically exchange control messages and establish shortcut paths. FTDC guarantees to deliver packets as long as the destination is reachable. The experiments show that network failures have minimal affection to the performance of FTDC.

References

[1] H. Abu-Libdeh et al. Symbiotic routing in future data centers. In *Proc. of ACM SIGCOMM*, 2010.
[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proc. of ACM SIGCOMM*, 2008.
[3] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 1953.
[4] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM CCR*, volume 34, pages 15–26. ACM, 2004.
[5] J. Dean. Designs, lessons and advice from building large distributed systems. *Keynote from LADIS*, 2009.
[6] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proc. of ACM SIGCOMM*, 2011.
[7] C. Guo et al. Dcell: a scalable and fault-tolerant network structure for data centers. In *Proc. of ACM SIGCOMM*, 2008.
[8] C. Guo et al. Bcube: a high performance, server-centric network architecture for modular data centers. In *Proc. of ACM SIGCOMM*, 2009.
[9] B. Karp and H. Kung. Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of ACM Mobicom*, 2000.
[10] S. S. Lam and C. Qian. Geographic Routing in d-dimensional Spaces with Guaranteed Delivery and Low Stretch. In *Proc. of ACM SIGMETRICS*, June 2011.
[11] V. Liu, D. Halperin, A. Krishnamurthy, and T. E. Anderson. F10: A fault-tolerant engineered network. In *Proc. of USENIX NSDI*, 2013.
[12] R. N. Mysore et al. Portland: a scalable fault-tolerant layer 2 data center network fabric. In *Proc. of ACM SIGCOMM*, 2009.
[13] C. Qian and S. Lam. ROME: Routing On Metropolitan-scale Ethernet . In *Proc. of IEEE ICNP*, 2012.
[14] J.-Y. Shin, B. Wong, and E. G. Sirer. Small-world datacenters. In *Proc. of ACM SOCC*, 2011.
[15] A. Singla, P. B. Godfrey, and A. Kolla. High throughput data center topology design. In *Proc. of USENIX NSDI*, 2014.
[16] A. Singla, C. Hong, L. Popa, and P. Godfrey. Jellyfish: Networking data centers randomly. In *Proc. of USENIX NSDI*, 2012.
[17] I. Stojmenovic. Position-based routing in ad hoc networks. *Communications Magazine, IEEE*, 40(7):128–134, 2002.
[18] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang. Adaptive forwarding in named data networking. *ACM SIGCOMM CCR*, 42(3):62–67, 2012.
[19] Y. Yu and C. Qian. Spaceshuffle: A scalable, flexible, and high-bandwidth data center network. In *Proc. of IEEE ICNP*, 2014.