Energy- and Priority-Aware Traffic Engineering for Content-Centric Networking

Ling Xu, Tomohiko Yagyu

Cloud System Research Lab
NEC Corporation, Japan
Email: `lingxu@gisp.nec.co.jp`, `yagyu@cp.jp.nec.com`

Abstract—**Content-Centric Networking (CCN) is a new network architecture aiming to solve many fundamental problems of IP networks. We are interested in the application of CCN carrier networks during natural disasters. During these times, carrier networks need to save energy until reinforcement arrives. Traffic delivery is prioritized, where high-priority traffic such as disaster news receives superior transmission quality. Multiple Tree-based Traffic Engineering (MTTE), a recently proposed traffic engineering scheme, realized the goal of energy-saving by forwarding traffic on minimal spanning trees. In addition, several priority-aware transmission schemes were proposed recently. However, no CCN communication scheme that realizes both energy-saving and prioritized routing has ever been considered. In this paper, we propose Priority-aware MTTE (PMTTE) to fill this vacuum. Given a certain energy consumption limitation, PMTTE optimizes the transmission quality of the high-priority traffic first. Then, it uses the remaining energy consumption quota to improve the low-priority traffic's quality. We compared the performance of PMTTE against priority queue-enabled MTTE (as a naive energy- and priority-aware CCN traffic engineering scheme). Simulation shows that compared with MTTE, PMTTE can improve the performance of high-priority traffic by up to 80%, and of low-priority traffic by more than 30%.**

Keywords–content-centric networking; priority-aware; energy efficiency; multiple trees

I. INTRODUCTION

A. Content-Centric Networking

Content-Centric Networking (CCN) is a network architecture that is proposed recently for solving IP architecture's problems [1]. Conventional IP networks focus on retrieving needed contents from certain hosts. Nowadays, however, users are only concerned about the contents, not from where to obtain them. In CCN networks, each piece of content is given a unique *name*. Users tell the networks the names of the needed contents. The networks independently identify hosts providing the contents and then fetch the contents to the users.

One of CCN's main features is that routers carry *caches* to reduce redundant traffic. Each host in the network may provide some contents and this host is called the *producer* of its contents. The producer splits each of its contents into a set of *chunks*. The producer also registers the name of each of its contents on each router - a process called *content publishing*. Hosts that request the content is called the content's *consumers*. When a consumer needs a piece content, it sends one request (called an *Interest*) for each chunk of the content. The Interest is then forwarded to the producer according to the information the producer registered on routers. When the producer receives an Interest, it sends the requested chunk back to the consumer. Each router along the chunk's forwarding route tries to store the chunk in its cache. The next time the router receives an Interest for the same content, if the chunk is still in its cache, the router replies directly with the chunk to the consumer.

We are interested in CCN's application in carrier networks during natural disasters and hope to make the CCN network both energy-efficient and priority-aware.

On one hand, networks consume a huge amount of energy, and improving network energy efficiency has been a popular research topic in recent years. In CCN networks, in-router caches cost extra energy. Hence, the networks generally consume more energy than conventional IP networks.

Xu et al. proposed *Multiple Tree-based Traffic Engineering* (MTTE) to reduce CCN's energy consumption [2]. Research has found that *network interfaces* (shortened as *faces* henceforth) of routers in modern networks are generally underutilized [3]. The idea of Xu et al. is to shut down as many faces of routers as possible. MTTE splits traffic on multiple tree-like networks that are generated based on the physical network. The trees are generated in such a way that the number of faces included in the trees is minimized. Since trees generally contain fewer faces than the original physical network, energy can be conserved.

On the other hand, real-world carrier networks deliver traffic of different *priorities*. During natural disasters, disaster alarms and news broadcasting are more important than entertainment TV programs. When disasters occur, the network traffic usually surges suddenly. MTTE is priority-agnostic and cannot ensure the performance of the high-priority traffic.

Several priority-aware transmission schemes have been proposed for CCN networks recently. However, to the best of our knowledge, none of the existing proposals realizes both energy- and priority-aware transmission. Through this paper, we hope to fill this vacuum.

B. Research Goal

In this paper, we assume that the network delivers two kinds of traffic: *high-priority* and *low-priority*. Examples of the high-priority traffic include live streaming such as disaster news. Such traffic is mission-critical and has to be delivered to users stably before certain deadlines. For examples, during the Great East Japan Earthquake [4], tsunami hit the seashore several minutes after the earthquake. If tsunami news can arrive at people in time, thousands of lives can be saved. The low-priority traffic can be other types of communication like binary file transmission. Compared with the high-priority traffic, low-priority traffic is less urgent and can be delayed or dropped.

The simplest idea to implement priority-aware transmission is to always deliver the high-priority traffic first using priority queues. By "using priority queues," we mean that each face contains one high-priority queue and one low-priority queue. High-priority (low-priority) packets are piled on the high-priority (low-priority) queue in a first-in and first-out manner. The face always sends packets in the high-priority queue before forwarding any packets in the low-priority queue. However, this approach would result in serious quality degradation of the low-priority traffic.

Through this research, we aim to create a traffic engineering scheme that provides these properties: (1) it reduces energy consumption of the whole network system; (2) given a upper bound for the system energy consumption and a lower bound for the transmission quality of the high-priority traffic, it guarantees both the bounds without greately sacrificing the quality of the low-priority traffic.

C. Proposal Summary

When a consumer requests a piece of content, we use *session* to denote the process in which the consumer receives all chunks. Each Interest belongs to one certain session. We call sessions transmitting high- (low-) priority contents the *high- (low-) priority sessions*.

Sessions that have at least one chunk not returned before a certain deadline ϕ_{dead} are regarded as *failed sessions*. We define *Session Failure Rate* (SFR) as the ratio of failed sessions among all sessions. Session failure rate is used as the main transmission quality metric in this paper.

We propose *Priority-aware MTTE* (PMTTE), and compare the performance of PMTTE with priority queue-enabled MTTE (as a naive priority and energy-aware CCN transmission scheme). Simulation shows that, compared with MTTE, PMTTE reduces the session failure rate of high-priority traffic by 80% and of low-traffic by 30%.

The rest of this article is organized as follows: Section II highlights relevant, prior literature in this field; Section III briefly introduces MTTE since MTTE is the base of PMTTE; Section IV introduces PMTTE's design and Section V evaluates PMTTE's performance; and Section VI is the conclusion.

II. Related Work

Several priority-aware transmission schemes have been proposed for CCN networks recently. However, to the best of our knowledge, no scheme that balances the trade-off between the network energy consumption and the transmission quality of prioritized traffic has been considered yet. Kim et al. proposed reserving bandwidth for the high-priority traffic [5] where their communication scheme is similar to the IntServ bandwidth reservation scheme for IP networks [6]. During natural disasters, important messages need to be delivered to more people. Routers have limited capacity and cannot forward all packets. Psaras et al. proposed that packets should be given names that specify their priorities [7]. Routers independently decide whether to forward a packet according to the routers' remaining capacities and the packet's priority. The objective of this research is orthogonal from ours. Immediately after a natural disaster, communication traffic may dramatically increase and overload the network. Psaras et al. proposed prioritizing contents according to the importance of the contents to the contents' consumers [8]. Somaya et al. used prioritized queues for delivering packets with different deadlines [9]. Traffic is classified into multiple groups. Routers maintain queues for accommodating different groups and packets with near deadlines are delivered first. Tsilopoulos et al. realized that for streaming contents in *Information-Centric Networking* (synonym of CCN), it is inefficient for consumers to request each chunk. They suggested that routers work as proxies that request content for consumers. [10]

Research effort has been made in assessing CCN's energy efficiency using simulation [11][12]. Some research compared the energy efficiency of CCN with existing IP-based content delivery techniques such as content delivery networking and peer-to-peer networking. Song et al. [13] noticed that in modern carrier networks, a great amount of traffic is generated from the edge. [13] uses GreenTE - an existing energy-aware TE mechanism designed for IP networks [3] - for reducing energy consumption of the core network, and uses CCN for eliminating redundant traffic generated by the edge network. However, Song et al.'s approach does not reduce the energy consumption of CCN itself.

III. Background Knowledge of MTTE

In CCN, each router contains a *Forwarding Information Base* (FIB). An FIB contains a set of entries and each entry is a mapping from one prefix to one or multiple router face(s). Suppose that the FIB contains two entries $fe0$=''/Asia/'':{face2} and $fe2$=''/Asia/Tokyo/'':{face2,face5} and that an incoming packet has a name "/Asia/Tokyo/music.mp3." The router finds the face whose FIB entry's prefix matches the packet's name and forwards the packets via this face.

A. Tree-Based Congestion Control

In MTTE, the network contains a central server called the *controller*. The controller maintains a database named the *tree set*. Initially, the controller creates one spanning tree based on the physical network topology and adds the tree into the tree set. The controller sends the tree set to routers. Routers update their respective FIBs so that they can forward the packets along the trees.

When congestion is about to occur, the controller creates more trees to spread the traffic and mitigate congestion. Specifically, routers periodically report the utilization of their faces to the controller. The controller calculates the *Congestion Rate* (CR) as the maximal value of the received utilization. When $CR > \phi_{tcc}$, where ϕ_{tcc} is a system parameter, the controller creates another tree and adds it to its tree set. This mechanism is termed as *Tree-Based Congestion Control* (TCC).

B. Face Weight Calculation

We use STs to denote the tree set in MTTE. Faces included in the tree set are called *live faces*, denoted by $E(STs)$. Faces not included the tree set are called *free faces*.

When the controller creates a new tree it calculates the weight of each face. Based on the weights, the controller runs Kruskal's minimal spanning tree algorithm. The controller then asks routers to evenly deliver traffic on trees in the new tree set.

To reduce energy consumption, the new tree should not dramatically increase the number of live faces. To this end, we call the live faces with utilization higher (lower) than ϕ_{ce} the *congested faces* (*uncongested faces*). The controller assigns weights to faces so that the *uncongested faces* are chosen first, free faces are chosen later, and congested faces are chosen last.

Namely, underutilized live faces are more likely to be used to create the new tree, which suppresses energy consumption.

A critical design in MTTE is that the diameters of the routing trees must be suppressed. Otherwise, when the diameters are large, not only is the transmission delay increased but also the cache utilization efficiency is decreased. Congestion is more likely to occur and the transmission quality deteriorates quickly.

MTTE uses a *Betweenness*-based heuristic algorithm to suppress the diameters of the routing trees. For each face e, the controller computes its "face betweenness" ($e.be$) - a widely used metric in graph theory [14]. Informally, $e.be$ represents the number of the shortest paths in the entire network that traverse e. Imagine that routers c and p are a pair of consumer and producer, and sp is the shortest path between c and p on G. Intuitively, if more faces with high betweennesses are added to st, the probability that the data transmitted on st between c and p are delivered along the shortest path is higher. Accordingly, the diameter of st will be small. Based on this observation, in MTTE, the controller selects faces with higher betweennesses first. When the controller creates the initial tree, for each face e, it calculates $e.eb$, and sets $e.weight = 1/e.eb$. When subsequent trees are created, the controller makes the weights of uncongested faces directly proportional to the faces' utilization and makes the weights of free faces inversely proportional to the faces' betweennesses. The complete tree creation algorithm is shown in Algorithm 1.

C. Hash-Based Traffic Splitting

Each time the controller changes the tree set, it asks routers to update their FIBs. First, the controller sends the tree set to all routers. Suppose that H is a collision-proof hash function preloaded on each router, and $N(d)$ is the name of content d. Routers update their FIBs so that d is forwarded on the $H(N(d))\%K$-th tree, where K is the size of the new tree set. After the update, packets are generally evenly delivered on each tree and the congestion can be mitigated.

D. Tree Removal

When $CR < \phi_{uu}$, where ϕ_{uu} is a preloaded system parameter, the controller removes the last tree in the tree set. The controller sends the new tree set to all routers, and asks routers to update their FIBs. Routers shut down their adjacent faces that are not included in the new tree set. Accordingly, energy consumption can be reduced.

IV. PMTTE: THE DESIGN

A. Problem Analysis

PMTTE uses priority queues-enabled MTTE as the base architecture. This architecture has two problems:

First, priority queues alone are not sufficient to ensure the transmission quality of the high-priority traffic. When the network has bandwidth bottlenecks, high-priority packets will drop anyway. MTTE works well in resolving bandwidth bottlenecks by creating more trees before congestion really occurs. Hence, in MTTE, the main reason that routers drop packets can be that the traffic changes so fast that no sufficient trees are created. Certainly we can accelerate tree creation by using smaller ϕ_{tcc}. However, creating more trees consumes more energy.

```
1:  $E_{UE}$ = Faces in $E(STs)$ with utilization $<= \phi_{ce}$.
2:  $E_{CE}$ = Faces in $E(STs)$ with utilization $> \phi_{ce}$.
3:  $E_{free}$: Faces not in $E(STs)$
4:  $U_{max}$ = maximum of face utilization of $E_{UE}$
5:
6:  The controller calculates the betweenness $e.eb$ of each face $e$.
7:
8:  if $|STs|$ = 0 then
9:      // The controller is creating the initial tree
10:     for all $e$ in the network do
11:         $e$.weight = $1/e.eb$
12:     end for
13: else
14:     // The controller creates a new tree for mitigating congestion
15:     for all $e \in E_{UE}$ do
16:         e.weight = e.utilization
17:     end for
18:     for all $e \in E_{free}$ do
19:         e.weight = $U_{max} + 1/e.eb$
20:     end for
21:     for all $e \in E_{CE}$ do
22:         e.weight = $U_{max} + 2$
23:     end for
24: end if
25:
26: The controller generates a minimal cost spanning tree $st$ using Kruskal's algorithm on the whole network.
27: if $st$ is different from all the existing trees then
28:     return $st$
29: else
30:     return FAILED
31: end if
```

Figure 1: Based on current face utilization, the controller generates a new spanning tree (Algorithm 1 from [2]).

Second, in MTTE, routing trees are created in such a way that they heavily overlap each other for reducing energy consumption. Since priority queues aggressively drop low-priority packets, the transmission quality of the low-priority traffic will be heavily impacted when congestion occurs.

We propose *Priority-Dependent Routing* (PDR) and *Face Separation* (FS) to address these problems. Algorithm 2 lists the PMTTE algorithm's work flow and functionalities.

B. Priority-Independent Routing

We assume that each packet of the high- (low-) priority traffic contains an attribute tag that specifies content' priorities, as done in [7]. Routers can know the priorities of incoming packets by checking priority tags.

To improve the transmission quality of the high-priority traffic, we propose PDR. PDR forwards the high- and low-priority traffic on independent trees. We use *whole traffic* to denote the total of the high- and low-priority traffic. We call the trees used for forwarding the high- (low-) priority traffic the *H (L) trees*. Analogously, MTTE only has one tree set (for the whole traffic) and we call it the *W trees*. We call the three parameters required by MTTE, ϕ_{tcc}, ϕ_{ce} and ϕ_{uu}, a *threshold set*. The controller holds independent threshold sets for the high- and low-priority traffic, respectively. The ϕ_{tcc} of the H trees is generally set lower than ϕ_{tcc} of the L trees. Hence, the high-priority traffic will have more sufficient routes and is less likely to get congested. Meanwhile, since we only accelerate the H trees' creation, energy consumption can be suppressed.

PDR improves the quality of the low-priority traffic as well. As traffic is separately forwarded, the congestion among traffic of different priorities in routers' forwarding queues can be mitigated. As a result, transmission quality of both the high- and low-priority traffic can be improved.

1) Implementation

To implement PDR in PMTTE, each router holds two FIBs - one for the high-priority traffic and the other for the low-priority traffic. When the system launches, the controller creates two spanning trees and adds them into the H trees and the L trees, respectively. These two trees are created using the original MTTE tree-creating algorithm. Each time the H trees (L trees) change, the controller sends the changed tree set to routers. When a router receives the H (L) trees, it updates its high- (low-) priority FIB using the similar traffic splitting algorithm of MTTE. The difference with MTTE is that, in MTTE, all prefixes (both high- and low-priority) are evenly split on the W trees. In PMTTE, the high- (low-) priority prefixes are evenly split over the H (or L) trees.

For each face e, we use $U_e(H)$, $U_e(L)$ and $U_e(W)$ to denote the utilization caused by the high-priority traffic, the low-priority traffic and the whole traffic, respectively. Similar to MTTE, routers in PMTTE periodically send the utilization of their faces to the controller. The difference between MTTE and PMTTE is that in MTTE, routers only send the $U_e(W)$. In contrast, in PMTTE, routers send both $U_e(H)$ and $U_e(W)$.

Based on the received utilization, the controller adds trees when congestion is about to occur (Algorithm 2). We use the *congestion rate of the high- (low-) priority traffic*, denoted as $\theta(H)$ and $\theta(L)$, as the maximum of the received $U_e(H)$ and $U_e(W)$. When $\theta(H)$ ($\theta(L)$) exceeds $\phi_{tcc}(H)$ ($\phi_{tcc}(L)$), the controller creates a tree for the H trees (L trees). Namely, the low-priority traffic flows in PMTTE in the same way the whole traffic flows in MTTE. Note that $\theta(L)$ is the maximum of $U_e(W)$ instead of $U_e(L)$. The latter means that when the controller creates a new tree for the L trees, the controller is more likely to pick faces used by the H trees. Later, when these faces get congested, low traffic's quality will seriously suffer.

2) Judgment of Live Faces

In the original tree creation algorithm, the controller needs to judge whether a face is live (Line 18, Algorithm 1). In MTTE, all faces used in the tree set are regarded as live faces. In contrast, in PMTTE, faces used in the H trees (rather than used in both the H trees and the L trees) are regarded as live faces when the controller creates a new tree for the high-priority tree set (Algorithm 6). The reason is as follows.

Recall that the transmission quality is closely affected by the diameters of the routing trees (Section III-B). Like in MTTE, when the controller in PMTTE creates a tree and needs to choose a free face, it chooses faces of higher betweennesses first in order to reduce the tree's diameter (Line 12, Algorithm 5). Accordingly, trees created later generally have larger diameters than trees created earlier.

The L trees are created based on the maximum of the whole traffic. Naturally, the number of trees in L tree set is likely to be larger than the number of trees in the H tree set. On average, trees in the L tree set are likely to have larger diameters than trees in the H tree set. By only using faces in the H tree set as live faces, PMTTE can suppress the diameter of the new high-priority tree.

3) Removing Underutilized Trees

If $\theta(H)$ (or $\theta(L)$) is lower than $\phi_{uu}(H)$ (or $\phi_{uu}(L)$), the controller removes the last tree from the H (or L) tree set.

C. Face Separation

Due to the overlapping nature of the tree creation algorithm of MTTE, the H trees and L trees would still heavily cover each other, which threatens the low-priority traffic's quality. To address this problem, when the controller creates a new low-priority tree, it increases the weights of faces that are used in the H trees by a factor of ϕ_{FS} (Algorithm 4). The controller can explicitly specify the extent that the L trees should disjoint from the H trees. This technique is called FS, and ϕ_{FS} is named the *Face Separation Rate*.

```
1: Creates the initial H tree;
2: Creates the initial L tree;
3: while $1$ do
4:    if $\theta(H) > \phi_{tcc}(H)$ then
5:       CreateTree(H)
6:    end if
7:    if $\theta(L) > \phi_{tcc}(L)$ then
8:       CreateTree(L)
9:    end if
10:   Sleeps for a certain period $T_{update}$;
11: end while
```

Figure 2: PMTTE work flow

```
1: $weights = \{\}$;
2: for all $e$ in the network do
3:    $weights[e] = GetFaceWeight(e, newTreeType)$;
4: end for
5: Calculates a Kruskal minimal spanning tree using
   $weights$;
6: Adds the new tree to the $newTreeType$ trees;
7: Sends the $newTreeType$ trees to routers for updating the
   FIBs;
```

Figure 3: The *CreateTree($newTreeType$)* method. $newTreeType$ is 'H' or 'L' when the new tree to create is for the H (or L) trees.

```
1: weight = GetDefaultWeight($e$, $newTreeType$)
2: if $newTreeType$ is $H$ then
3:    return weight
4: else
5:    return $weight \cdot \phi_{FS}$
6: end if
```

Figure 4: The *GetFaceWeight(e, $newTreeType$)* method for FS. The controller calculates the weight of a face e when creates a new tree.

D. Energy Consumption and Transmission Quality Trade-off

We use session failure rate (defined in Section I-C) as the metric for transmission quality, and use *Live Face Rate* (LER) as the metric for energy consumption. We use SFR(H) and SFR(L) to denote the SFR of the high- and low-priority traffic, respectively. LER is defined as $\gamma/|E|$, where γ is the number

```
if IsLiveFace($e$, $newTreeType$) then
    $u$ = GetUtilization($e$, $newTreeType$);
    // $e$ is an uncongested face
    if $u < \phi_{ce}(newTreeType)$ then
        $weight = u$;
    else
        // $e$ is a congested face
        $weight = u + 2$;
    end if
else
    // $e$ is a free face
    $weight = \phi_{ce}(newTreeType)$ + 1/e.betweenness;
end if
return $weight$;
```

Figure 5: The *GetDefaultWeight(e, $newTreeType$)* method.

```
if newTreeType is H then
    return $e$ is used in H trees;
else
    return $e$ is used in H trees or L trees;
end if
```

Figure 6: The *IsLiveFace(e, $newTreeType$)* method

```
while 1 do
    if $LER > \phi_{energy}$ then
        if $\phi_{tcc}(L) < 1.0$ then
            Increases $\phi_{tcc}(L)$;
        else
            if $\phi_{tcc}(H) < 1.0$ then
                Increases $\phi_{tcc}(H)$;
            end if
        end if
    else
        if $SFR(H) > \phi_{sfr}$ then
            if $\phi_{tcc}(H) > 0$ then
                Reduces $\phi_{tcc}(H)$;
            end if
        else
            if $\phi_{tcc}(L) > \phi_{uu}$ then
                Reduces $\phi_{tcc}(L)$;
            end if
        end if
    end if
    Sleeps for $T_{update}$;
end while
```

Figure 8: Energy consumption and transmission quality trade-off

of faces used in the L and H trees, and $|E|$ is the total number of faces in the physical network.

During disasters, network administrators determine a upper bound, ϕ_{energy}, for the system energy consumption and a lower bound, ϕ_{sfr}, for the transmission quality of the high-priority traffic. The administrators dynamically adjust $\phi_{tcc}(H)$ and $\phi_{tcc}(L)$ in a heuristic manner (Algorithm 8). When real system energy consumption exceeds ϕ_{energy}, $\phi_{tcc}(L)$ or $\phi_{tcc}(H)$ is increased to reduce the system energy consumption. When $LER <= \phi_{energy}$ and $SFR(H) > \phi_{sfr}$, $\phi_{tcc}(H)$ is reduced to improve the high-priority traffic's quality. When $LER <= \phi_{energy}$ and $SFR(H) < \phi_{sfr}$, $\phi_{tcc}(L)$ is reduced to improve the low-priority traffic's performance. In this way, administrators can guarantee the requestments for energy consumption and the high-priority traffic, and minimize the low-priority traffic's quality degradation.

V. Evaluation

This section evaluates PMTTE's performance by comparing PMTTE with priority queue-enabled MTTE (MTTE).

A. Topology

Our simulation is performed on ndnSIM – a simulation platform developed by UCLA for CCN-related research [15].

The simulation runs on the network topology of autonomous system 3257 (AS3257). This topology is provided in Rocketfuel network dataset [16], a dataset that has been used in network research [17][18]. Each node in AS3257 represents a router. We extract the largest connected component of AS3257 and use all the remaining nodes for creating trees. AS3257 contains three types of routers: *cores*, *gateways* and *leaves*.

```
if $newTreeType$ is $H$ then
    return $U_e(H)$;
end if
return $U_e(W)$;
```

Figure 7: The *GetUtilization(e, $newTreeType$)* method.

According to the definition of Rocketfuel datasets, leaves are the routers with degrees equal to or less than two, gateways are the routers directly connected to the leaves, and the remaining routers are cores.

We assume that in real-world CCN networks, consumers are adjacent to leaves; low-priority producers, such as servers of entertainment applications, are adjacent to gateways and leaves; high-priority producers, such as servers of national TV stations, connect to cores. We assign one low-priority producer to each leaf and gateway, and assign one high-priority producer to each core. In total, 132 low-priority producers and 108 high-priority producers are located. Each producer provides 20 different pieces of content.

B. Simulation Parameters

We assign $ConsumersPerNode$ high- and low-priority consumers to each leaf. Each high- (low-) priority consumer randomly chooses one high- (low-) priority content from the whole high- (low-) priority prefix pool. End consumers start requesting selected content after certain delays so that network traffic volume follows a sine wave shape (for simulating the traffic spike during the disaster).

We change $ConsumersPerNode$ to control the maximal network traffic volume. Two volumes, with $ConsumersPerNode$ being 60 and 80, are evaluated. The number of high-priority consumers is 4,800 (60 x 80) and 6,400 (80 x 80), respectively. Similarly, the number of low-priority consumers is 4,800 and 6,400 as well. Each consumer issues 10 Interests per second (10 ips), meaning that maximally 10 chunks should be returned by certain producers per second. The size of each chunk is 1024 bytes. According to the simulation, the average number of hops between each pair of consumer and producer is approximately 10. Hence, maximally 983 MB (4,800 x 2 x 10 ips x 10 hops x 1024 bytes) traffic can flow within the network per second when $ConsumersPerNode$ = 60. Meanwhile, each face has a capacity of 5 Mbps. The network contains 420 faces.

The theoretical capacity of the whole network is 260 MBps (5 Mbps x 420 / 8). Therefore, we believe that the network is sufficiently congested to evaluate the effectiveness of the proposed algorithm when $ConsumersPerNode = 60$ and 80.

The threshold set of the L trees in PMTTE is equal to the threshold set of the W trees in MTTE. Both sets have $\phi_{tcc} = 0.8$, $\phi_{ce} = 0.6$ and $\phi_{uu} = 0.5$.

The number of high- and low-priority consumers is equal. Intuitively, the high- and low-priority traffic have equal volumes. As ϕ_{tcc} of MTTEs' W trees is 0.8, ϕ_{tcc} of PMTTE's H trees should be equal or less than half of it, i.e., 0.4. For PMTTE's H trees, we set the default values of ϕ_{tcc}, ϕ_{ce} and ϕ_{uu} to 0.4, 0.3 and 0.15, respectively.

C. Performance Metrics

We evaluate the performance using SFR and LER. Session failure rate is directly decided by the session failure deadline ϕ_{dead}. ϕ_{dead} should be set according to the mean round-trip time (RTT) of the system. The RTT in different communication systems may vary greatly. In [19], the authors evaluated the Quality of Experience (QoE) of a streaming system and they assumed a RTT of 150ms. Streaming applications generally request low RTTs. In this paper, we are considering general carrier networks where both streaming and non-streaming traffic exist. Hence, the RTT in our system should be larger than the 150ms. In our simulation, we set ϕ_{dead} to 200ms.

Recall that LER is defined as $\gamma/|E|$. For MTTE, γ is the average number of faces used in the whole tree set during the simulation, For PMTTE, γ is the average number of faces used in the L and H trees during the simulation.

We expect to see that, compared with MTTE, PMTTE further reduces SFR(H) and SFR(L) at the price of a moderate increase in LER. The effectiveness of PDR and FS are evaluated, respectively.

We repeat each evaluation scenario ten times and display the mean values of each performance metrics.

Parameters used in the simulation are summarized in Table I.

TABLE I: Simulation parameters

Parameter	Value
Total number of faces	420
Total number of routers	240
Number of gateway routers	52
Number of leave routers	80
Number of core routers	108
Prefixes per producer	20
Chunk size	1024 bytes
Cache size	2000 chunks
(For PMTTE) Default H ϕ_{tcc}, ϕ_{ce}, ϕ_{uu}	0.4, 0,3, 0.15
(For PMTTE) Default L ϕ_{tcc}, ϕ_{ce}, ϕ_{uu}	0.8, 0.6, 0.5
(For MTTE) Default Whole ϕ_{tcc}, ϕ_{ce}, ϕ_{uu}	0.8, 0.6, 0.5
Duration of one simulation run	50 seconds
Update interval T_{update}	2 seconds
High-priority producer location	cores
Low-priority producer location	gateways, leaves
Session failure deadline ϕ_{dead}	0.2 seconds

D. Evaluating Priority-Independent Routing

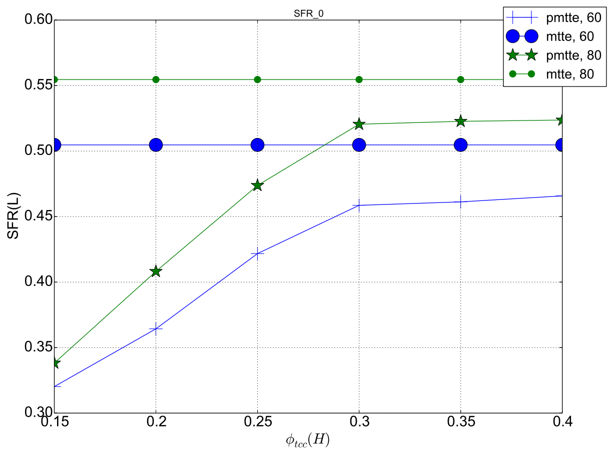

Figure 9: Session failure rate for the low priority traffic. Legends indicate the algorithm and traffic volume used for evaluation. For example, "pmtte, 60" indicates that PMTTE is evaluated when $ConsumersPerNode = 60$.

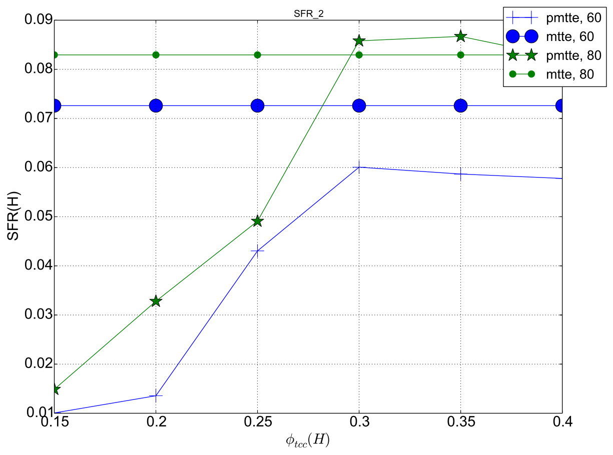

Figure 10: Session failure rate for the high priority traffic.

Figures 9 and 10 compare the session failure rates of the low-priority traffic and high-priority traffic between PMTTE and MTTE. On the Figures, $\phi_{tcc}(H)$ increases from 0.15 to 0.4 on the x-axis. For evaluation displayed in this section, ϕ_{FS} is set to 1.0 (i.e., FS is disabled).

[19] shows that in video streaming systems, the users' QoE degrades rapidly when the packet drop rate exceeds certain thresholds. Since we consider prioritized transmission, instead of limiting the absolute value of the session failure rate, it is more important to focus on the performance superiority of the high-priority traffic over the low-priority traffic. In this paper, we assume that SFR(H) should be lower than 2% to guarantee decent QoE. Figure 10 shows that when $\phi_{tcc}(H)$ is 0.15, SFR(H) can be reduced to lower than 2%.

As $\phi_{tcc}(H)$ decreases, more H trees are created and SFR(H) of PMTTE is suppressed. In Figure 10, it is notable that PDR effectively reduces SFR(H) when $\phi_{tcc}(H)$ is reduced to 0.15. SFR(H) reduces from 7.2% to 1% when

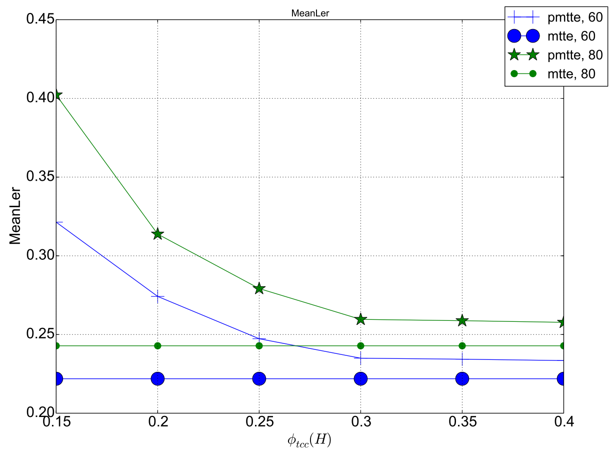

Figure 11: Live face rate.

$ConsumersPerNode = 60$, and reduces from 8.3% to 1.5% when $ConsumersPerNode = 80$. Meanwhile, when SFR(H) is less than 2%, SFR(H) can be than more than 10 times lower than SFR(L) (Figure 9 and Figure 10).

Meanwhile, as expected in Section IV-B, PDR remarkably reduces SFR(L) as well (Figure 9), as compared with MTTE. When $ConsumersPerNode = 60$, SFR(L) decreases from 51% to 33%. When $ConsumersPerNode = 80$, SFR(L) decreases from 56% to 34%.

Figure 11 plots the influence of PDR on energy consumption. Clearly, the improvement in transmission quality is at the price of higher energy consumption. Even though PMTTE still suppresses LER to be lower than 40%. In real-world network systems, the network operator can tune $\phi_{tcc}(H)$ to make the trade-off between transmission quality and energy consumption.

E. Evaluating Face Separation

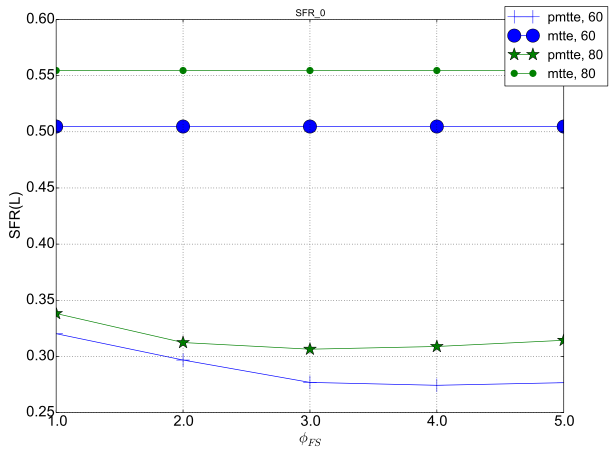

Figure 12: Session failure rate for the low priority traffic. Legends indicate the algorithm and traffic volume used for evaluation. For example, “pmtte, 60” indicates that PMTTE is evaluated when $ConsumersPerNode = 60$. (FS enabled)

Figures 12 and 13 compare the session failure rates between PMTTE and MTTE when FS is enabled. The results are

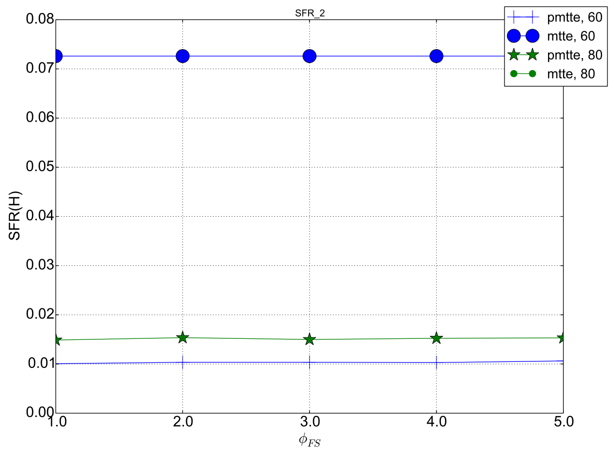

Figure 13: Session failure rate for the high priority traffic. (FS enabled)

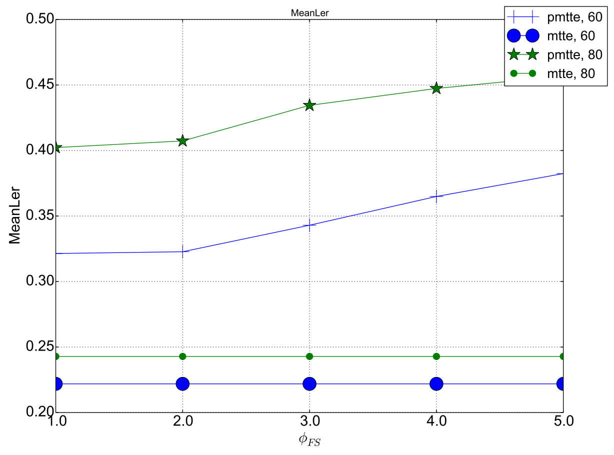

Figure 14: Live face rate. (FS enabled)

obtained with $\phi_{tcc}(H) = 0.15$ (the value obtained in Section V-D where SFR(H) of PMTTE falls below 2%). The face separating rate ϕ_{FS} increases from 1.0 to 5.0 on the x-axis. When ϕ_{FS} takes 1.0, the results are equal to those obtained when FS is disabled and PDR only is enabled.

Figures 12 and 13 clearly reveal that priority queues deteriorate the low-priority traffic. Meanwhile, FS can effectively mitigate this deterioration. As the face separation rate increases, SFR(L) decreases (33% to 27% for $ConsumersPerNode = 60$, and 34% to 32% for $ConsumersPerNode = 80$). Moreover, FS does not affect SFR(H) (Figure 13), meaning that network operators can improve the performance of the low and high traffic using PDR and FS independently.

Figure 14 shows that FS moderately increases energy consumption. However, LER is still lower than 50% in PMTTE.

VI. Conclusion and Future Work

Through this research, we aim to create a traffic engineering scheme that is both energy- and priority-aware for CCN networks. The challenge is to avoid heavily impacting the transmission quality of the low-priority traffic. Our proposal, PMTTE, splits the high- and low-priority traffic on separate routing trees. This enables us to flexibly increase the tree recreation frequency for the high-priority traffic to suppress the high-priority traffic congestion. At the same time, this change mitigates the collision between the high- and low-priority traffic, which improves the quality of both. Last but not least, by aggressively separating the routes of the high- and low-priority traffic, we can actively improve the quality of the low-priority traffic. Simulation using a real-world ISP network topology shows that compared with naive priority queue-enabled MTTE, PMTTE can boost the quality of both the high- and low-priority traffic by up to 50%. As for future work, we plan to improve the transmission quality of the high-priority traffic by creating routing trees that minimize the distances between the high-priority producers and consumers. Meanwhile, MTTE is a centralized scheme and will fail when the connection between routers and the control is cut. We plan to make MTTE more fault-tolerant, especially during disasters.

Acknowledgment

The work described in this paper was, in part, performed in the context of the FP7/NICT EU-JAPAN GreenICN project.

References

[1] L. Zhang et al., “Named data networking (NDN) project,” Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC, 2010.

[2] L. Xu and T. Yagyu, “Multiple-tree based online traffic engineering for energy efficient content centric networking,” in Proc. The Sixth International Conference on Advances in Future Internet (AFIN2014). IARIA, 2014, pp. 121–127.

[3] M. Zhang, C. Yi, B. Liu, and B. Zhang, “GreenTE: Power-aware traffic engineering,” in Network Protocols (ICNP), 2010 18th IEEE International Conference on. IEEE, 2010, pp. 21–30.

[4] N. Mimura, K. Yasuhara, S. Kawagoe, H. Yokoki, and S. Kazama, “Damage from the great east japan earthquake and tsunami-a quick report,” Mitigation and Adaptation Strategies for Global Change, vol. 16, no. 7, 2011, pp. 803–818.

[5] Y. Kim, Y. Kim, and I. Yeom, “Differentiated services in named-data networking,” in Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on. IEEE, 2014, pp. 452–457.

[6] B. Braden, D. Clark, and S. Shenker, “Integrated services in the internet architecture: an overview,” Internet Requests for Comments, RFC Editor, RFC 1633, June 1994, http://www.rfc-editor.org/rfc/rfc1633.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1633.txt

[7] I. Psaras, L. Saino, M. Arumaithurai, K. Ramakrishnan, and G. Pavlou, “Name-based replication priorities in disaster cases,” in Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on. IEEE, 2014, pp. 434–439.

[8] G. Tyson, E. Bodanese, J. Bigham, and A. Mauthe, “Beyond content delivery: Can icns help emergency scenarios?” Network, IEEE, vol. 28, no. 3, 2014, pp. 44–49.

[9] S. Arianfar, P. Sarolahti, and J. Ott, “Deadline-based resource management for information-centric networks,” in Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013, pp. 49–54.

[10] C. Tsilopoulos and G. Xylomenos, “Supporting diverse traffic types in information centric networks,” in Proceedings of the ACM SIGCOMM workshop on Information-centric networking. ACM, 2011, pp. 13–18.

[11] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, “In-network caching effect on optimal energy consumption in content-centric networking,” in 2012 IEEE International Conference on Communications (ICC). IEEE, 2012, pp. 2889–2894.

[12] U. Lee, I. Rimac, D. Kilper, and V. Hilt, “Toward energy-efficient content dissemination,” Network, IEEE, vol. 25, no. 2, 2011, pp. 14–19.

[13] Y. Song, M. Liu, and Y. Wang, “Power-aware traffic engineering with named data networking,” in Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN), 2011. IEEE, 2011, pp. 289–296.

[14] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” Proceedings of the National Academy of Sciences, vol. 99, no. 12, 2002, pp. 7821–7826.

[15] A. Afanasyev, I. Moiseenko, and L. Zhang, “ndnsim: Ndn simulator for ns-3,” Named Data Networking (NDN) Project, Tech. Rep. NDN-0005, Rev, vol. 2, 2012.

[16] “Rocketfuel dataset,” https://github.com/cawka/ndnSIM-ddos-interest-flooding/tree/master/topologies/rocketfuel_maps_cch, (retrieved: July, 2015).

[17] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with rocketfuel,” ACM SIGCOMM Computer Communication Review, vol. 32, no. 4, 2002, pp. 133–145.

[18] C. Yi et al., “A case for stateful forwarding plane,” Computer Communications, 2013, pp. 779–791.

[19] T. N. Minhas, O. Gonzalez Lagunas, P. Arlos, and M. Fiedler, “Mobile video sensitivity to packet loss and packet delay variation in terms of qoe,” in Packet Video Workshop (PV), 2012 19th International. IEEE, 2012, pp. 83–88.