CGMコンテンツに適した
タグキャッシングルータアーキテクチャの研究

黒瀬 浩

博士（情報学）

総合研究大学院大学

複合科学研究科

情報学専攻

平成24年度（2012）

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士（情報学）授与の要件として提出した博士論文である.

審査委員:

（主査）　山田 茂樹　国立情報学研究所／総合研究大学院大学
福田 健介　国立情報学研究所／総合研究大学院大学
計 宇生　国立情報学研究所／総合研究大学院大学
鯉渕 道紘　国立情報学研究所／総合研究大学院大学
奥田 隆史　愛知県立大学大学院

（主査以外はアルファベット順）

Tag Caching Router Architecture with Folksonomies for CGM Contents

Hiroshi Kurose

Doctor of Philosophy

Department of Informatics,

School of Multidisciplinary Sciences,

The Graduate University for Advanced Studies (SOKENDAI)

March, 2013

A dissertation submitted to

the Department of Informatics,

School of Multidisciplinary Sciences,

The Graduate University for Advanced Studies (SOKENDAI)

in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

Advisory Committee：

Shigeki Yamada (Chair)	National Institute of Informatics / The Graduate University for Advanced Studies (SOKENDAI)
Kensuke Fukuda	National Institute of Informatics / The Graduate University for Advanced Studies (SOKENDAI)
Yusheng Ji	National Institute of Informatics / The Graduate University for Advanced Studies (SOKENDAI)
Michihiro Koibuchi	National Institute of Informatics / The Graduate University for Advanced Studies (SOKENDAI)
Takashi Okuda	Aichi Prefectural University

(Alphabetic order of last name except chair)

論文要旨

本論文では，CGM（Consumer Generated Media）の動画視聴に適したコンテンツ配信基盤であるタグキャッシングルータ（TCR: Tag Caching Router）アーキテクチャを提案する．TCR アーキテクチャは，フォークソノミータグをキーとして，コンテンツ検索およびコンテンツ情報およびコンテンツデータのキャッシングを行ない，適した位置からコンテンツデータを取得する．このため，ネットワーク全体のトラヒックを削減し，ネットワーク負荷の変動による視聴中断を防止することができる．

Web2.0 が普及して視聴者が容易にコンテンツを投稿できるようになり動画投稿サイトの利用が促進された．投稿者が増えるにつれコンテンツ数，視聴者とも増加している．加えて，動画品質の向上から単位時間あたりのデータ量も増加し続けている．また，SNS（Social Networking Service）が普及してコンテンツの評判が急速に視聴者間へ伝搬されることが多くなった．その結果，ネットワークトラヒックまたは，CMS（Content Management System）サーバ負荷の増加，コンテンツ検索精度の低下が発生している．

TCR アーキテクチャは，以下 3 点を前提として動画コンテンツ視聴を安定的に行うコンテンツ配信基盤を提供する． 1）人気コンテンツのアクセス分布は Zipf の法則に従う，2）動画検索・視聴はフォークソノミータグを利用することが多い， 3）CGM コンテンツのランキングは時間的変動が大きい．

TCR アーキテクチャの特徴は， 1) 通信経路上のルータにコンテンツの情報およびデータをキャッシュする， 2) フォークソノミータグによりコンテンツ拡散および検索を行なう， 3) 各 TCR は他 TCR の状態を意識せずに適した通信経路を形成する，の 3 点である．TCR アーキテクチャの効果は，シミュレーションにより国内に CMS サーバがあることを想定したネットワーク環境での平均ダウンロード時間で確認する．評価は，キャッシュを用いない伝統的な方法，URI (Uniform Resource Identifier）によるキャッシュ方法およびタグによるキャッシュ方法を比較し TCR アーキテクチャが一番優れていることを示す．

評価方法は，タグをキーとしてキャッシュする方法の先行研究が無い事から以下のように， 1) 視聴端末と CMS サーバの間に 1 つのキャッシュルータを持つタンデム構成， 2)

CMS サーバが集中配置され視聴端末が CMS サーバと近い位置にある格子状構成， 3) タグアクセスランキングを考慮した国内 CMS へのツリー状構成， 4) 先行研究で評価されている階層キャッシュの登録方法の影響， 5) 応答性，安定性および経済性からなる総合評価，の 5 段階から確認する．

国内にある CMS サーバを用いてコンテンツ視聴を行うことを想定したルータ 100 台規模のネットワークシミュレーションでは，URI を検索キーとしたコンテンツキャッシング方式に対して，タグをキーとした TCR では，4 から 12%のキャッシュサイズでも同等の応答性および安定性が得られ，評価指標による総合評価においても優れていることを示す．

TCR アーキテクチャが CGM 動画視聴に適した FIA（Future Internet Architecture）であることを示す．TCR アーキテクチャは，FIA，特に CCN（Content-Centric Networking）で提案されているアーキテクチャにおいて CGM 動画視聴に特化するため以下の前提を除外することで単純かつ効率的なコンテンツ配信基盤を提供する． 1) コンテンツの名前解決とコンテンツデータ保持は別システムにより管理される， 2) コンテンツが任意の場所から生成され保持される， 3) コンテンツを一意に特定しコンテンツデータを取得する．

現実的な CGM 動画視聴環境と評価環境の相違点は，現実的にはタグ数およびコンテンツ数が評価環境より大きいが，タグランキング分布は Zipf 状となると想定されることから，TCR に実装するキャッシュ容量をスケールアップすることにより TCR アーキテクチャは現実モデルに適応できると考える．

将来課題として，TCR アーキテクチャの性能は，コンテンツに割り当てられるタグ数，コンテンツ数，人気コンテンツの評判の伝搬速度により強く依存すると考えられるため，タグアクセスの時間的変動も含め，より現実的なアクセス状況での評価が必要である．

TCR アーキテクチャは，既存のルータ，SNS または視聴端末への実装が容易であり，既存のインターネットに少数の TCR ルータから導入可能である．また FIA, CCN のルーティングまたは名前解決機構との組み合わせによる統合の可能性が得られた．

Abstract

This paper proposes a tag caching router (TCR) architecture that supports folksonomies(tag)-based search and content caching for consumer-generated media (CGM) content. TCR architecture propose the architecture of the Tag Caching Router (TCR) to cache content information and content to the router on the communication path used as a search key folksonomy tags. To retrieve contents from the point at which it was appropriate, thereby reducing overall network traffic, we watch to prevent interruptions due to fluctuations in network load in this TCR network.

The TCR architecture assumes the following characteristics of CGM video viewing. 1) Tag access ranking distribution is subjected by the Zipf's law. 2) Almost viewers of CGM content use folksonomies for finding target contents. 3) Temporal variation of ranking of CGM content that has been posted by the viewers is change dynamically.

TCR architecture is characterized by the following three points. 1) TCRs on the communication path can cache information and data content. 2) Each TCR can advertise content information and search content using the folksonomies. 3) Each TCR works searching contents or caching contents without state of other TCRs.

In order to confirm the effect of the TCR architecture, we have evaluated the performance by the average download time in a networked environment is assumed to create the simulator for CMS servers in Japan domestic.

Evaluation results were compared among traditional method(without cache), cached contents by URIs, and cached contents with tags. The result of tag-based caching is most excellent on this environment.

In order that there is no previous research on how to cache as tags. We had plan to concrete evaluation method in four stages. 1) Tandem configuration: Caching router between viewer's terminals and CMS servers. This case is very simple. 2) Lattice topology: Routers are connected to a grid-like. Viewer's terminals are connected to a router of the outer edge of the grid.

CMS servers are located in some of the inner grid routers. 3) Tree-like topology: Access ranking survey by tags on tree-like topology. 4) Comprehensive evaluation contains three factors such as response, stability, and economic efficiency. Also, how to put for cache hierarchy in previous studies for assumed network was evaluated on the tree-like topology.

The simulation results indicate that the TCR architecture can achieve the equal download time and download time variation to those of a caching architecture with a URI-based search while the TCR architecture requires 4 to 12% of the cache capacity of the URI-based caching architecture.

TCR architecture is explained that it is the architecture that specializes in video viewing CGM. TCR architecture provides a simple and efficient content delivery platform that specializes in video viewing CGM by omitting the following assumptions of the proposed architecture in Future Internet Architecture (FIA). 1) Name resolution and data content that retention of the content is managed by another system. 2) Be stored content is generated from any location. 3) Viewers can get the content data to uniquely identify the content.

Difference in the evaluation environment and realistic video viewing environment was mentioned. The number of tags and content on the evaluation environment is sufficiently smaller than the real model. However, We believe that the distribution of tag access similar the Zipf distribution. Therefore, the real model can be achieved by scaling up the capacity of the cache of each TCR routers of evaluation model.

Future challenges of this study is to evaluate the performance of the TCR architecture access conditions more realistic. It is possible for the reason, the network performance is strongly depended by the propagation speed on social network, the number of attached tags to the content, and the number of content.

TCR architecture is expected to be combined with both name resolution function and routing of FIA and/or CCN technologies.

目 次

本論文中で使われているサービス・システム・製品名は，一般に各社・各組織の商標または登録商標であり，内容は執筆時点の情報を基にしている．

図目次

表 目 次

略語

BA	Barabási-Albert (Network topology model)
CID	Content IDentifier
CCN	Content-Centric Networking
CDN	Content Delivery Network
CGM	Consumer Generated Media
CNF	Cache aNd Forward
DNS	Domain Name System
DTN	Delay/Disruption Tolerant Network
FIA	Future Internet Architecture
FIFO	First-In, First-Out
FTP	File Transfer Protocol
FQDN	Fully Qualified Domain Name
GFID	Global File IDentifier
HTTP	Hyper Text Transport Protocol
ICP	Intetnet Caching Protocol
ISP	Intetnet Service Provider
KVS	Key Value Store
LCD	Leave Copy Down
LCE	Leave Copy Everywhere
LFU	Least Frequently Used
LRU	Least Recently Used
MCD	Move Copy Down
MTU	Maximum Transmission Unit
NDN	Named Data Network
OSI	Open Systems Interconnection
P2P	Peer to Peer
QoE	Quality of Sercvice
QoS	Quality of Experience
RSS	RDF site summary (1.0), really simple syndication (2.0)
SLA	Service Level Agreement
SLB	Server Load Balancing
SE	Search Engine
SNS	Social Networking Service
TCP/IP	Transmission Control Protocol/Internet Protocol
TCR	Tag Caching Router
URI	Uniform Resource Identifier

用語

フォークソノミー

フォークソノミー **(folksonomies)**	協働で分類すること
タグ **(Folksonomy Tag)**	フォークソノミーで分類するための語句
ロングテイル **(Long Tail)**	下位ランクの参照数が低く長く分布すること

検索

検索結果数	検索で得られたアイテム数
適合文書数	検索結果のうち検索者が許容するアイテム数
正解数	検索対象全体での正解アイテム数
適合率 **(precision)**	正確性の指標．適合文書数 / 検索結果数
再現率 **(recall)**	網羅性の指標．検索結果数 / 全体の正解数
検索性能 **(F**$_{measure}$**)**	検索性能の指標．適合率と再現率の調和平均

CCN（Content-Centric Networking）

Core Router / コアルータ	ネットワークトポロジーで内部側のルータ
Edge Router / エッジルータ	視聴端末が接続されるネットワーク端のルータ

ノードの種類

CMS (Content Management System)	コンテンツの登録・検索・視聴を提供するサービス
SNS (Social Networking Service)	視聴者間でコンテンツの情報を共有するサービス
Viewer	視聴者がコンテンツを検索・視聴するための端末
SE (Search Engine)	語句からコンテンツリストを返す検索エンジン
DNS (Domain Name System)	FQDN から CMS の IP アドレスを返す名前変換サーバ

TCR（Tag Caching Router）アーキテクチャ

アドバタイズメントフェーズ	視聴候補コンテンツリストをネットワークに拡散する段階
コンテンツ検索フェーズ	視聴者が視聴候補コンテンツを取得し視聴対象を特定する段階
ダウンロードフェーズ	視聴するコンテンツを視聴端末に転送する段階
コンテンツリゾルバ	要求を受信した TCR が転送先を特定する名前解決機能
キャッシングフィルタ	各 TCR が受信したパケットからキャッシュするための機構
キャッシングポリシ	キャッシングフィルタが自身にキャッシュするための判断基準
タグ割当数 $ntag$	1 つのコンテンツに付与するタグの数の平均
キャッシュ率 $cache\%$	ルータのキャッシュ総量と総コンテンツサイズ総量の比
タグ検索率 $tagsearch\%$	タグによる検索と全検索数の比
タグ総数 N	対象とするタグの総数
トップ **K** 位 K	タグアクセス頻度がロングテールの裾野となる順位
割引率 d	タグのアクセスランキング分布のロングテールの程度
タグランキング分布 $D_{k,d,K}$	本評価で用いる順位 k のアクセス頻度

（意図的な空白ページ）

第1章　序論

インターネットの普及により利用できるサービスが増加して利用形態も変化している．Web2.0 技術 [1] が普及した 2005 年以降は，利用者によるコンテンツの動的生成と利用者間の交流が盛んになり，Web2.0 に対応したサービスや技術が進展している．特に動画検索や SNS（Social Networking Service）では，フォークソノミータグを用いた分類・検索が用いられるようになった．

フォークソノミー [2] とは，利用者がコンテンツに関連する語句をコンテンツに付与して視聴者間で共有することによりコンテンツ探索を容易にする分類方法である．付与する語や句をフォークソノミータグと呼ぶ．本論文ではフォークソノミータグを単にタグと記す．

本論文では，動画投稿サイトに投稿された動画を視聴する際のコンテンツ配信基盤に着目し，既存のシステムおよび技術の問題点を解決する TCR（Tag Caching Router）アーキテクチャを提案してシミュレーションにより効果を測定する．

本章では，1.1 節で研究の概要を述べ，1.2 節でソーシャルメディアの変化および配信基盤の特性から研究の背景を確認し，1.3 節で提案する TCR アーキテクチャのコンセプトを提示し，1.4 節でまとめる．本章の最後で本章以降の構成を提示する．本論文は，著者の研究 [3–5] を基にしている．

1.1　研究概要

1.1.1　研究の目的

本論文では，視聴対象コンテンツが視聴者の評判により急変することの多い CGM 動画のダウンロードトラヒックをタグをキーとして経路上にキャッシュすることで，ネットワーク全体のトラヒックを削減する方法を検討する．その効果をシミュレーションにより定量的に評価することを目的とする．

1.1.2 従来技術の問題点

従来のコンテンツ配信基盤は，ネットワーク負荷が増加してもダウンロード速度の低下を抑制する方法を追求している．また，新たなインターネットの利用方法からアーキテクチャを見直している FIA（Future Internet Architecture）でも，利用形態に応じて適切な位置からコンテンツデータを取得する経路探索機能が検討されている．

しかし，これらのコンテンツ探索は，コンテンツを一意に特定する方法であり，動画視聴で用いられるフォークソノミーと連動していない．近年，CMS（Content Management System）利用者の増加や SNS による急速な評判の伝搬が発生するため，コンテンツ配信基盤にもタグによる検索や経路選択が求められる．

CGM（Consumer Generated Media）や報道動画などは，多くの派生コンテンツが制作されるため，視聴者は，特定のコンテンツを探索するよりも関心のある動画を視聴できれば満足することが多い．タグによるコンテンツ検索は，所望のコンテンツを見つけ出す有効な方法となる．タグのアクセスランキングは，スケールフリー性 [6] に従うことが知られており [7]，上位ランクのタグを持つコンテンツをキャッシュすることにより少ないストレージコストで同等以上の応答性を実現することが期待できる．

この方法は，動画投稿サイトなどの CMS で集中管理しているランキング計算のコストを低減し，ランキング反映までの遅延時間を短縮する効果も備える．

既存のコンテンツ配信基盤の問題は，第 2 章で確認する．

1.1.3 研究のアプローチ

本論文では，以下の CGM 動画視聴で顕著な特性を前提とする．

1. 人気コンテンツのアクセス分布は，Zipf の法則 [8] に従う
2. 動画検索・視聴はタグを利用することが多い
3. 視聴者が投稿した CGM コンテンツのランキングは時間的変動が大きい

提案する TCR アーキテクチャは，以下の特徴を持つ．

1. コンテンツの情報およびデータを通信経路上のルータでキャッシュする
2. タグによりコンテンツ拡散および検索を行なう
3. 各 TCR は，他 TCR の状態を意識せずに適した通信経路を形成できる

国内に CMS サーバがあることを想定したネットワーク環境における平均ダウンロード時間による性能評価を，キャッシュを用いない伝統的な方式，URI (Uniform Resource Identifier）による

キャッシュ方式，および，タグをキーとしてキャッシュする TCR 方式で比較する．タグランキング分布は，アクセスモデルからロングテールの裾野の高さと長さにより，平均ダウンロード時間の影響を確認する．

評価方法は，先行研究が無いことからシミュレータを作成し，以下のように段階的に行い，性能および問題点を確認しながら進める．

1. 視聴端末と CMS サーバの間に 1 つのキャッシュルータを持つタンデム構成
2. CMS サーバが集中配置され，視聴端末がサーバと近い位置にある格子状構成
3. ツリー状構成でタグアクセスランキングの影響
4. 先行研究で評価されている階層キャッシュの登録方法の影響
5. 応答性，安定性，および経済性からなる総合評価

1.1.4 研究による貢献

タグをキーとした経路上のキャッシュ方法は研究されておらず，本論文により基本的なアーキテクチャ，性能，および課題を提示する．この方式は，特定のコンテンツを取得しなくても良い動画視聴においてコンテンツ探索方法を提示し，増加し続けるネットワーク負荷を低減する．ネットワークプロバイダやコンテンツの視聴機会を増やしたいニーズを持つ CMS 提供者に対して新たな情報流通基盤の方法を提供しうる．

1.2 研究の背景

1.2.1 ソーシャルメディアの変化

Web2.0 によりインターネット上のコンテンツを利用者が動的に更新できるようになった．従来，Web サイトの管理者は，HTML（HyperText Markup Language）を用いて制作したコンテンツを公開し，E メールで受け付けた利用者の要望を次回の Web ページ更新時に反映していた．Web2.0 の普及以降は，動的にコンテンツを生成する技術により利用者の投稿したコンテンツから動的に Web ページが生成できるようになり，コンテンツの種類に応じて多彩なサービスサイトが開設された．コンテンツの量および質が高いサイトには，更に利用者および投稿者が増えて，利用者に応じたサービスの向上が図られ，分野毎に代表的なサイトが現れた．

動的にコンテンツを生成するためのツールとして，複数の利用者が協働で物事のまとめを行う CMS ツールである PukiWiki [9] や Plone [10], 無料で Wiki サービスを提供する Wiki サイト [11] が現

表 1.1: インターネットで普及しているメディアの定性的属性

種類	同時性	遅延許容	再使用	データサイズ	寿命[a]
CGM 動画	不要	中	多い	数 M〜数 GB	数日〜数年
商用動画	不要	厳しい	少ない	数百〜数 GB	数月〜数年
自習教材	不要	中	多い	数十 M〜数 GB	数年
講義教材	要	中	少ない	数十 M〜数 GB	数年
ネットニュース	不要	中	中	数十 M	数日
ネットラジオ	不要	中	中	数十 MB	数日
ビデオ会議	要	厳しい	無し	特定不可	その時のみ
ストリーミング	要	厳しい	無し	特定不可	その時のみ
メール	不要	緩い	無し	数 MB	その時のみ
ファイル転送	不要	緩い	少ない〜多い	〜数 GB	数月〜数年

[a]コンテンツが一定時間内に一定のアクセス数を持つことを寿命範囲とする

れた．コンテンツ別の代表的ポータルサービスとして，辞書は WikiPedia [12]，動画は Youtube [13]，ニコニコ動画 [14]，音楽は last.fm [15]，写真は Flickr [16]，イラストは Pixiv [17] などがある．

コンテンツが豊富になると利用者が関心のあるコンテンツを容易に見つけられるように検索エンジンの精度向上（pageRank [18] やコンテンツ周辺の文字列をコンテンツに関連するとしてキーワード検索できるようにする）や知識の共有 SocialBookMark [19] が利用された．特に利用者間の情報交換を可能とするソーシャルネットワークサービス（SNS）が盛んになり，時間・空間を超えたコミュニケーションが盛んになった．この例に Blog，mixi [20]，Twitter [21] などがある．

コンテンツを特徴から分類・検索する方法は，コンテンツの種類および量が膨大となったため，従来，コンテンツ制作者がカテゴリを決め，区分していたディクショナリ方式では，対応できなくなった．そのためコンテンツに特徴を表す語句を付け，その語句を利用者間で共有し改善していくフォークソノミー（Folksonomies）が普及した．

学術論文や文書管理では，後で検索しやすいように著者や管理者がキーワードを数個付けることを行ってきたが，Web2.0 以降では，自動付与や利用者による動的変更が行われるようになった．

1.2.2 配信基盤の特性

対象とするサービスの特性

表 1.1 にインターネットで普及しているメディアの定性的な特性を示す．メディアの種類により同時性，遅延許容，再使用，データサイズ，およびコンテンツの寿命に違いがある．本論文で

表 1.2: データ転送の制約事項

用途	順序[a]	単位[b]	利用制約	途中での蓄積	切断時の処理
ファイル転送	R	B	データ取得完了後使用	可能	蓄積場所の探索
動画再生	S	F	取得速度 > 再生速度	可能	蓄積場所の探索
Web カメラ	S	F	同時性（緩い）	可能	キャッシュ転送先
ユビキタス	R	R	同時性（緩い）	可能	転送先探索
IP 電話	S	F	同時性（厳しい）	ほぼ不可能	再接続
中継配信	S	F	遅延許容時間	遅延制約内で可	再接続

[a]転送順序 R:Random, S:Sequential
[b]転送単位 B:Block, F:Frame, R:Record

対象とする CGM 動画は，コンテンツ供給側と視聴側の同時性は必要とせず，コンテンツの再利用が多く，商用の動画サービスよりビットレートに厳しくないことが多い．また，他のサービスと比べてデータサイズが大きく，コンテンツの寿命に幅がある．商用動画のアクセス数は，宣伝と当時にアクセスが急増して，その後，急速に減少するが，CGM 動画は，SNS により評判が伝搬されるためアクセス数が減った後でも増加することがあり，コンテンツアクセス数の時間的変動予測が難しい．

表 1.2 にデータ転送の制約事項を示す．用途により利用時の制約事項がある. 例えば，ファイル転送では，データの同一性を確保するためチェックサムの照合を行うので，データを取得し終えないと利用することができない．同時性を重視する IP 電話では，転送遅延が発生した際に，転送ビットレート低速化やフレーム廃棄により同時性を確保する機能を保有している．ユビキタスネットワークでは，全てのノードからのデータ受信を前提とせず取得できたデータの範囲内で最大のサービスを提供する方法を採る．

動画視聴では，データ取得と再生を並行して行うために，データ取得速度は，再生速度より高速であることが求められる．CMS から視聴端末までの通信経路上の各リンクにおいて速度差がある場合は，スループットが最低のリンクにより性能が制限される．最低速度のリンクが予め判明する場合は，そのリンクの視聴端末側にキャッシュを設けることでダウンロード性能を改善することができる．

視聴者の振舞い

表 1.3 にコンテンツ視聴のフェーズ別技術を示す．従来の放送型メディアおよび Web2.0 が出現するまでの静的コンテンツでは，提供者と視聴者が明確に区別され，視聴者のフィードバックを次のコンテンツ制作に反映するまで数週間から数ヶ月を要していたが，Web2.0 によりインターネッ

表 1.3: 動画視聴のフェーズ別利用技術と評価指標

フェーズ	宣伝 →	検索 →	データ取得 →	視聴 →	協働
主体	制作者	視聴者	配信基盤	視聴者	
主要技術	Web2.0: SNS,Blog, RSS,CMS	Ranking: SE,CMS, RSS,WiKi	配信基盤: HTTP, CDN,P2P	プレイヤ: mp4 等, ブラウザ	ブラウザ: CMS, SNS
評価指標	アクセス率	再現率, 適合率	通信速度	QoS, QoE	再視聴率 行動率
提案技術	主体者主導	リコメンド	キャッシング	主体者主導	リコメンド

ト上のサイトに視聴者がコンテンツを動的に投稿できるようになり，視聴者・供給者・制作者の区別があいまいになった．その結果，各自ができることを行いコンテンツを制作・普及する協働作業が促進された．CGM 動画では，他者の投稿動画をもとに加工・編集して派生コンテンツを制作することや，SNS により次回の作品制作の支援や宣伝が行われインターネット上のコミュニティが形成されるとともにそれらを支援するツールが発展した．

様々な立場のインターネット利用者が多数のコンテンツを生成すると，従来のカテゴリや階層によるディレクトリ型情報分類（taxsonomy）では，分類の区分作成が間に合わず，検索精度（適合率）が低下した．この問題は，コンテンツの特徴を表す語句をコンテンツに付与してその語句を協働で追加・編集・削除することでコンテンツを検索するフォークソノミー (Folksonomy) 型の分類方法により解決された．

1.3 TCR（Tag Caching Router）アーキテクチャのコンセプト

本論文では，視聴者がタグをキーとして検索でき，動画コンテンツデータの順序性を確保して視聴者端末から取得しやすい位置からデータを取得する TCR を提案する．

1.3.1 利用イメージ

図 1.1 に TCR の利用イメージを示す．タグによるキャッシングとルーティングを行う複数の TCR が存在するネットワークに，コンテンツのオリジナルデータを持つ CMS，コンテンツの評判を保持・拡散する SNS，コンテンツを視聴する視聴端末が接続される．CMS または SNS は推奨するコンテンツのリストをネットワークに提供する（PUSH）．視聴端末からは視聴コンテンツのタグにより検索を行うとネットワーク内のルータのルーティングとキャッシングの機能により適

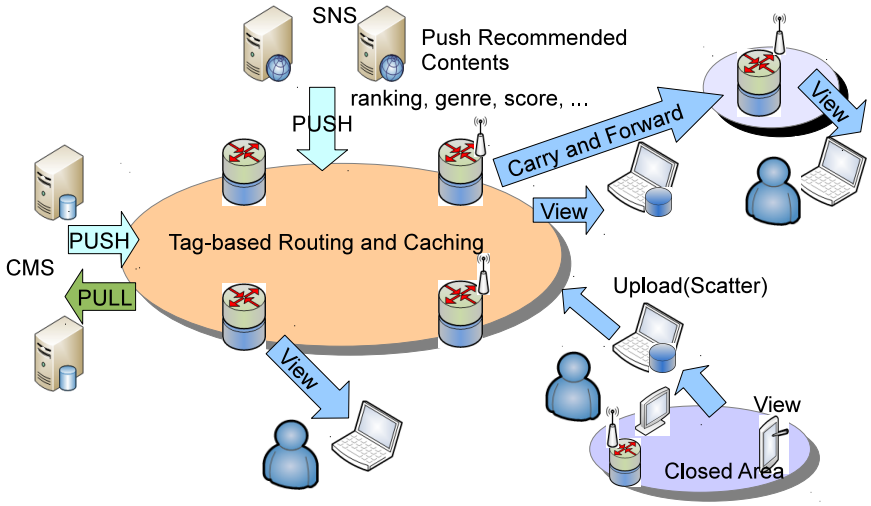


図 1.1: TCR 利用イメージ

した位置より視聴候補コンテンツやコンテンツデータを取得できる．また，視聴端末は，キャッシュと経路探索機能を持ちモバイル端末として動作することで，限定的な視聴を主とするネットワークと別のネットワークの間でコンテンツの評判とデータを伝搬する．

このネットワークアーキテクチャは，コンテンツの種類および量が増大し，視聴者の嗜好が多様となり，かつコンテンツの人気が時間的に変動する次世代のネットワークを指向する．

TCR の主機能を以下に述べる．

PUSH CMS または SNS は，視聴者へ推奨するコンテンツを TCR ネットワークに送信する．その情報を受信した TCR は，他の TCR へ情報を送信することで，視聴者の検索・ダウンロード要求に対して利用される機会を増加させる．

PULL 視聴端末の検索要求またはダウンロード要求によりネットワーク上の各 TCR はコンテンツの情報またはデータを探し出し視聴端末に結果を返す機能を持つ．この時，通信経路上の TCR にコンテンツ情報またはデータをキャッシュすることで，次の要求に対して経路長を短縮する．

Scatter 視聴端末が視聴したコンテンツを他のネットワークに接続した際にコンテンツの情報とデータを送信することでコンテンツ情報の拡散を行う．

Carry and Forward 視聴端末で視聴したコンテンツを別のネットワークで視聴者が推奨し，コンテンツデータと評判を視聴者が伝搬する．

Closed Area での利用 商店街や美術館など特定の地域や分野のコンテンツを Carry and Forward に

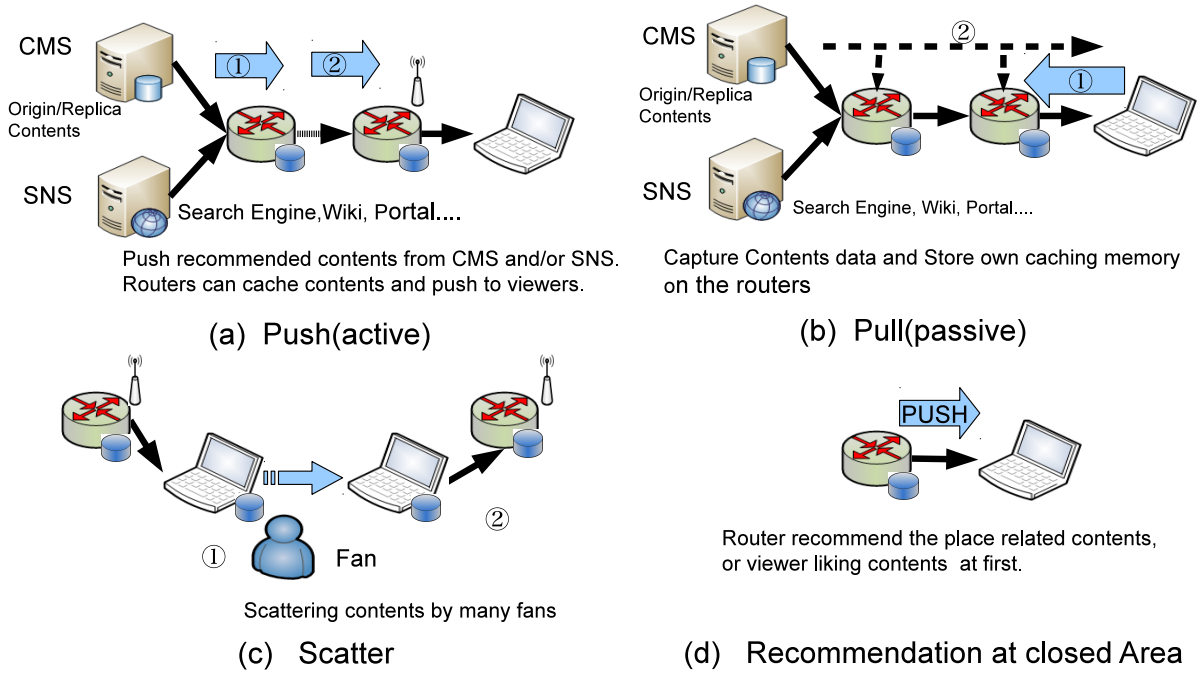


図 1.2: TCR 機能イメージ

より視聴者がコンテンツのデータと評判を伝搬する．

図 1.2 に TCR の機能イメージを示す．図 1.2 では図 1.1 で使用している機能要素について説明する．

(a) Push TCR は，CMS または SNS から推奨コンテンツのリストを受信すると自身にキャッシュするか判断し，コンテンツリストの内容により転送先を決定して，推奨コンテンツを転送する．

(b) Pull TCR は，視聴端末から発行されたコンテンツ検索要求またはダウンロード要求を受信すると，情報取得のための経路探索を行ない要求を転送する．また，その応答を受信すると，自身にキャッシュするか判断してキャッシュし，応答を視聴端末に向けて転送する．

(c) Scatter 図 1.1 の Carry and Forward を実現するために，TCR の Push, Pull の機能を持った視聴端末が別のネットワークに接続した際に，視聴者により推奨コンテンツの情報またはデータを Push することで評判とコンテンツを伝搬する．

(d) Recommendation at Closed Area 図 1.1 の Satter 機能を実現するために，Closed Area 内の CMS に Push 機能を設け，コンテンツの拡散を促進する．

第 3 章以降では，ルータの主機能である Push と Pull に着目し検討を行う．

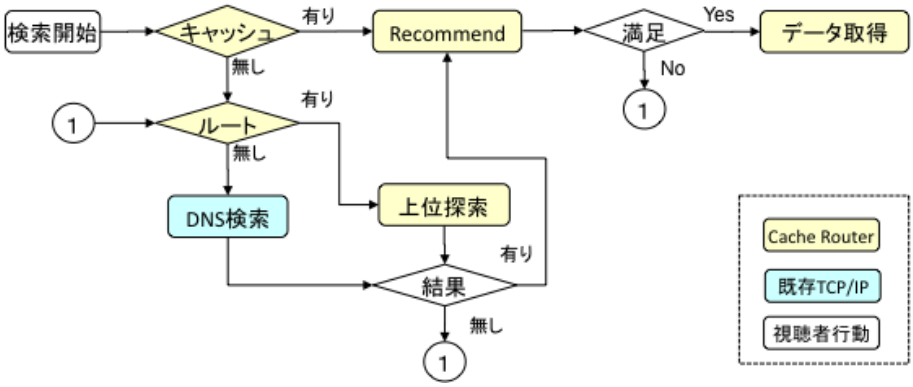


図 1.3: TCR：コンテンツ検索処理の概略

1.3.2 コンテンツ検索処理の概要

図 1.3 にコンテンツ検索処理の概略を示す．視聴者は，関心のあるコンテンツをタグを指定して検索する．要求を受信したルータは，内部のキャッシュを確認し関連情報があれば視聴回数の多いコンテンツ数件を推奨コンテンツリストとして返す（Recommend）．視聴者が推奨コンテンツリストから視聴するコンテンツを決定すると，視聴端末はダウンロード要求を発行し，その要求を受信したルータはコンテンツデータのダウンロードを行う．

一方，ルータのキャッシュに関連情報が存在しない場合は，ルータは，コンテンツ検索の照会先経路をキャッシュから検索して，存在すれば，該当するルータを選定して検索要求を転送する（上位検索）．キャッシュに経路が無ければ，従来手法である検索エンジンと DNS を用いてコンテンツ関連情報を取得する．

視聴者が検索結果の推奨コンテンツに満足しない場合または検索結果が無い場合は，TCR は，経路選定を行い直して再検索する．

以上からタグをキーとして複数のルータが検索要求を処理することで，コンテンツ情報と転送先経路をキャッシュから取得するネットワークを形成する．

1.3.3 リコメンド処理

数百万件のコンテンツから数件の推奨コンテンツの選択する方法は，視聴者の検索結果の満足度を左右する．既存手法と TCR の方法の定性的相違を表 1.4 に示す．

会員サイトなどで，会員が好むコンテンツを提示する方法に personalise がある． personalise では，各自の行動履歴から会員の嗜好を記録する．

表 1.4: リソース集計方法の相違

技術 / 項目	personalise	Social bookmark	TCR
目的	嗜好から推奨	コミュニティ形成	コンテンツ選択の容易化
情報更新	各自の行動	グループ内の行動	検索要求・応答
リポジトリ	集中	集中	分散
変数	$Tag, Resource$	$User, Tag, Resource$	$Tag, Resource, Router$
同期／非同期	同期	同期	非同期
更新契機	集計時	集計時	検索・ダウンロード時

他者の行動履歴から嗜好を共有する方法に Social Bookmark がある．この方法では自分の嗜好に合ったモデルとなる人を探すことで行動履歴の蓄積時間が短縮でき，自分の嗜好に近い人達がコミュニティを形成することで，検索精度を向上できる．

上記の方法では，自分に近い嗜好の人を探すために，サイトで類似度の集計を行う必要がある．集計は，会員（User），コンテンツ（Resource），コンテンツを特徴づけるタグ（Tag）それぞれにおいて類似度を計算する必要がある．人気のある SNS ほど，User, Resource, Tag の数は指数的に増加するため，類似度の集計に多大な計算が必要となる [7].

動画投稿サイトでは集計時間と現状の状況を近づけるために頻繁に集計を行うことから計算資源の増強が必要になる．コンテンツは増加しつづけるため，顧客が許容する検索精度を確保するための計算資源のコストは増大し続ける．計算量を減らすためには，ランキングの上位のみを集計対象とする方法があるが，CGM コンテンツは，ロングテールの傾向が強いためランキングが下がるほど検索精度が下がると考える．

この問題を解決するために，通信経路上の各 TCR ルータが推奨コンテンツのリストとデータを保持することで集中的な計算を行わなくても済む．

1.4 まとめ

人気の急変する動画コンテンツのダウンロード集中によるトラヒック増加を低減してデータサイズの大きなコンテンツデータのダウンロードにおいてダウンロード速度が再生速度を上回るコンテンツ配信基盤が求められる．視聴者が動画コンテンツの検索にタグを使用する機会が増えているため，コンテンツ配信基盤は，タグによる検索機能が求められる．

検索精度を高めるためにアクセスによる順位付けが必要であるが，集中的な集計方式は，計算資源が増大し続けるため，資源を増強せずに状況に応じた検索精度を保つ機能が求められる．

以上から，通信経路上にアクセス状況に応じて推奨コンテンツリストとデータをキャッシュでき，タグによるコンテンツ検索機能を提供するコンテンツ配信基盤の検討が必要である．

CMS でのランキング集計は，ユーザー間，コンテンツ間，タグ間およびそれらの関連性を集計しているため，データ数が大きくなると多大な計算が必要となる．一方，TCR では，タグのみに着目しても視聴者からの検索・視聴を経路上のルータでキャッシュし，かつキャッシュ廃棄方式を LRU(Least Recentry Used) を用いることにより多大な計算を必要としない．また，CMS では，ランキング変動の頻度により集計頻度を状況により変化させる必要があるが，TCR では，調整機構は不要である．

1.5 本論文の構成

本論文の構成を示す．第 1 章では，タグキャッシングルータの必要性と概要を述べた．続く，第 2 章で関連する先行研究と事例により課題を再確認し，第 3 章で提案システムのアーキテクチャ詳細について述べる．第 4 章で評価のために作成したシミュレータの構造と評価条件を，第 5 章で結果と分析を，第 6 章で全体のまとめを記載する．付録 A では，シミュレータの使用資源測定結果を示す．

（意図的な空白ページ）

第2章　関連する研究・技術

本章では，本研究に関連する先行研究および技術について述べる．初めに，2.1 節で実用されている技術・サービスとの関連を，次に，2.2 節でタグをキーとした検索の課題を，2.4 節で新たなインターネットアーキテクチャの研究動向を，2.5 節で動画視聴に特化した場合の課題を述べる．2.6 節では，タグをキーとしたコンテンツ配信基盤の要素技術を述べてまとめる．

本研究は，次世代インターネットアーキテクチャ（FIA: Future Internet Architecture）の一部であり CCN（Content-Centric Networking）の研究分野で動画視聴に特化しフォークソノミータグをキーとしたコンテンツ配信基盤基盤を指向している．このため，他の CCN, FIA 技術との組み合せによる利用が期待できる．

2.1　従来技術との関連

インターネットの普及によりホームページのアクセスが増え，サーバやネットワークの負荷増加によりホームページ取得に時間を要することから，Proxy サーバによるキャッシュが行われた．これらは，回線速度が低く，インターネット接続料金が従量制の場合には十分機能した．特に画像ファイルのキャッシュはホームページの取得時間の短縮に貢献した．1997 年に Proxy サーバを利用するためのプロトコル ICP （Internet Cache Protocol）[22]，AICP（Application of Internet Cache Protocol）[23] が規定され，並列動作とキャッシュ方法の改善の研究がなされた [24–26].

コンテンツ供給者は，利用者がホームページの再訪問を頻繁に行うようにホームページの変更を頻繁に行うことを迫られた．利用者の増加とともにネットワーク回線の光ファイバ化による回線速度向上や HTTP サーバのコンテンツを複製して複数のサーバによる負荷分散 [27–29] が行われるようになり，ネットワーク回線側とサーバ側の双方で利用者増加による応答時間を改善する多様な技術が導入された．

コンテンツ提供者は，コンテンツを頻繁に提供することがサイトの利用者増加に結びつくことがわかり，コンテンツを動的に生成する CMS（Content Management System）ツールが開発された．CMS の例としては PukiWiki [9], Plone [10], WordPress [30] などがある．

2005 年以降，インターネット上で動的にコンテンツを生成できるサービスを開始するサイトが現れると，従来，コンテンツを視聴するのみであった利用者がコンテンツを投稿するようになり，多くのコンテンツサイトが登場してサービス対象毎に分科した．例を上げると百科事典：WikiPedia [12]，写真：Flickr [16]，音楽：Last.fm [15]，動画：YouTube [13]，ニコニコ動画 [14]，画像：PiXiv [17]，学術会議日程：WikiCFP [31] などがある．これらは，同じ関心を持つ利用者がコンテンツ投稿やサイト運営に参加するため，コンテンツの質または量がある程度揃うと，利用者が急増してコンテンツの質および量は，大幅に向上された．

コンテンツの数が多くなると，利用者が求めるコンテンツを容易に検索できるように検索機能も改善された．検索エンジン側で提供していた区分を階層的に探索するディレクトリ型は，利用者による多様なコンテンツの表現説明により機能しなくなった．HTML 文書内で画像の周辺にあるキーワードから画像検索を行う機能が現れた．検索エンジン Google [32] ではリンク元とリンク先から各ページのスコアを計算して検索精度を高めている [18]．しかし，この方法もコンテンツとキーワードが直接結びついていないため検索精度に不満が残った．

上記問題を解決する方法として，コンテンツ制作者，コンテンツ供給者に加えてコンテンツ利用者がコンテンツに適した語句をコンテンツに付与するフォークソノミーが利用された．フォークソノミーは，コンテンツの特徴を表す語句（フォークソノミータグ）を用いてコンテンツを分類する方法で，ディレクトリ型のタクソノミーと対をなす．

音楽コンテンツを検索する場合には，演奏者，ジャンル，音楽タイトル，または，制作会社などを用いた検索方法が確立しているが，動画はコンテンツの特徴を語句で表すフォークソノミーが有効である．特に利用者が制作する CGM では，制作会社やジャンルが決められないコンテンツも多い．ニコニコ動画ではタグ検索およびタグ編集機能を提供している [14]．

コンテンツ制作会社が提供する商用コンテンツに加えて一般利用者が制作する Blog や CGM による投稿が増えると，各利用者が情報を入手するより，同様の関心を持つ利用者の情報を利用することで検索コストを削減できるようになった．Twitter [21] のつぶやきをキーワードで検索する Buzztter [33] や有名人や著者の関係を図示するスパイシー [34] は，容易に関連コンテンツを検索できることから利用者が多い．

個人が有用と考える情報を他者に公開し情報を共有するソーシャルブックマーク [35] の利用が促進された，サービスサイトの例として，Delicious [36]，はてなブックマーク [37]，citeUlike [38] などがある．

利用者に人気の有るコンテンツを容易に選択させる方法として，ランキング，リコメンデーション，タグクラウド [39–41] などがある．

インターネットは，無線 LAN によるモバイルデバイスの対応を行ってきたが，インターネット

開発当初に想定していない膨大なコンテンツ数，アクセス場所の動的変動，電話・放送・データ通信の融合などの新たなニーズに対応するため，新たなフレームワークを必要としているため，次世代ネットワーク （FIA：Future Internet Architecture）の研究が盛んになった．

2.2 フォークソノミーを用いたコンテンツ検索の課題

Web2.0 により動的コンテンツが生成されるようになると，検索にフォークソノミー，ソーシャルブックマーク，リコメンデーションが重視される．これらの先行研究について確認する．

フォークソノミー $\boldsymbol{F}$ は，タグ $\boldsymbol{T}$，利用者 $\boldsymbol{U}$, コンテンツリソース $\boldsymbol{R}$ と各要素の関係 $\boldsymbol{Y}$ を用いて $\boldsymbol{F} = (\boldsymbol{T}, \boldsymbol{U}, \boldsymbol{R}, \boldsymbol{Y})$ で表す [42]．各要素間の関係は，一度に求まらず，$(\boldsymbol{T}, \boldsymbol{U})$, $(\boldsymbol{U}, \boldsymbol{R})$, $(\boldsymbol{R}, \boldsymbol{T})$ と 3 種の関係を求める必要がある．また自身の嗜好に近いコミュニティを見つけるためには，各要素について類似度を求める必要がある．

タグによる検索で，利用者の満足度を上げるために，flickr, delicio.us のタグをクラスタリングして検索エンジンや Wikipedia を参考情報として関連性を生成する方法が提案されている [43]．ニコニコ動画 [14] では，2013 年 1 月時点で 865 万以上の動画が投稿されており，完全なフォークソノミーを求めるためには，膨大な計算資源を必要とする．タグ，ユーザ，リソースとも日々増加しているため，検索精度を保つためには，集計頻度の調整が必要となる．

計算量を削減する方法として，タグ・ユーザ間とタグ・リソース間の 2 段階の計算で評価した例では，文献情報管理サイトの集計対象は，12,000 ユーザ，16,000 タグ，論文 83,000 報であった [44]．

動画コンテンツの評判は，SNS を通じて急速に伝搬するため，CMS サイトの集計頻度を短くしないと検索精度を高めることができない．

検索結果の正確性を表す指標は，適合率（precision）と再現率（recall）が用いられる [45]．検索対象全体を Ω，検索結果で得られたアイテム数を N，その中で結果が満足する件数を R，Ω の中で満足する件数を C とすると，

$$\text{適合率}\, precision = \frac{R}{N} \quad where\ R, N \subseteq \Omega \tag{2.1}$$

$$\text{再現率}\, recall = \frac{R}{C} \quad where\ R, C \subseteq \Omega \tag{2.2}$$

$$\text{検索性能}\, F_{measure} = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2R}{N + C} \tag{2.3}$$

で表される．一般に適合率，再現率の一方を上げると他方が下がる傾向にある. CMS サイト bibsonomy, Delicious のタグについて調べた文献 [46] でも同様の結果が得られている．Web コンテンツでは，アクセスランク順の分布がロングテール状になることが知られているが，タグの場合も

同様の傾向がある [7]．タグのアクセス分布は，冪乗則や Zipf [8] で近似される．Web サイトの文書の分布 [8] や YouTube のジャンル別の投稿数 [47] は，Zipf 状である．

フォークソノミーを短時間で集計するには，ロングテールの裾野を除外し，ランキング上位のタグを対象にする方法が有効である．FolkRank [48] では，式 2.2, 式 2.3 を $tag(\boldsymbol{U}, \boldsymbol{R})$ に適用し，リソースに割り当てるタグ数が 4 で再現率 60%を，タグ数 8 で再現率 80%の結果であった．以上から上位数件のタグにより利用者アクセスの大多数が検索結果に満足すると考える．

数多くのタグからユーザに推奨するタグをリコメンドする方法が検討されている [19]．適合率を高めるため類似度を用いてリコメンドコンテンツを提示する方法 Social Ranking [44] では，適合率を利用者のタグ付け頻度とコンテンツの人気で評価している．

ニコニコ動画では，タグをクリックするとタグに関する百科事典”ニコニコ大百科” [49] を参照でき，タグの意味，そのタグが利用された背景，関連コンテンツ，掲示板，関連商品を参照・投稿でき，タグまたはコンテンツとの関連付けを促進する機能を持つ．

CGM 動画コンテンツを対象としたコンテンツ配信基盤は，以下の特徴から改善できる．

1. 視聴者は，自身の視聴経験から関心のあるタグを知っている場合が多いためユーザ類似性を除外することで計算量を削減できる．

2. 経路上で集計を行うことにより CMS での一局集中の集計から，タグとリソースによる局所的な集計により集計対象を減らせランキングの反映時間を大幅に短縮できる．

3. タグアクセスランキング分布の Zipf 分布から上位ランクのコンテンツをユーザ近傍のルータにキャッシュすることでダウンロード性能およびストレージコストを改善できる．

この方法を採用すると，SNS によりコンテンツの評判が急速に伝搬する場合に対応でき，人気のあるコンテンツほど短い通信経路でコンテンツデータを取得することが期待できる．

2.3 従来のコンテンツ配信基盤の課題

コンテンツダウンロード時間を短縮する方法として，従来より Proxy キャッシュ，P2P（Peer-to-Peer），CDN（Content Delivery Network）がある．

これらは，多彩なコンテンツが大量に制作され，データサイズが大きく，ランキング順位によりアクセス数の差が大きな CGM 動画コンテンツの配信基盤に向いていない．

コンテンツ配信基盤では，コンテンツを探索するための名前解決と経路選択を行う機構に分類できる．

2.3.1 Proxy Cache

古典的な Proxy キャッシュは，組織内のサーバにキャッシュするため，不特定多数が視聴する CGM 動画コンテンツのキャッシュには向かない．

CMS サーバ側でのコンテンツ複製を含んだ Proxy では，サーバ数とその配置によりダウンロード速度と経済性にトレードオフの関係が生じる．文献 [50] では，アクセス頻度により更新間隔を調整してサービス性と経済性を調整する機能を提示している．ネットワーク上の通信負荷によるボトルネックを解消するためには，多くの複製先が必要となり複製コストが高くなる．

2.3.2 P2P（Peer-to-Peer）

フォークソノミーを利用したコンテンツ配信方式は, P2P (Peer-to-Peer) にも応用されている [51]. P2P（Peer-to-Peer）では，ユーザ側に名前解決と経路選択の機構を持つ．参加者の増加によりコンテンツの取得機会が増えるが，経路長短縮は考慮されない．データ量が数 GB のサイズになる Linux ディストリビューションの配布には，サーバの負荷集中を避けるため Bittorrent [52] が利用されている．Bittorrent では，データを固定長の断片で管理し，見つけた断片から取得する方式のため，全データを取得し終えるまでデータを利用できない．動画視聴では，データはシーケンシャルに取得しないと再生中断が発生する可能性があるため，P2P は動画ファイルの配信基盤として向かない．

モバイルピアで人気によるコンテンツの取得速度を評価した文献 [53] では，人気のあるファイルの取得は 10 秒以内が 60%程度，不人気の場合は 40%程度であり，動画視聴のように視聴対象が多い場合は，不人気コンテンツが大多数を占める分布となるため，大多数の視聴者において取得時間の長時間化が予想される．

2.3.3 CDN（Content Delivery Network）

CDN（Content Delivery Network）では，ネットワークエッジ部に名前解決と経路選択の機構を持ちキャッシュ更新は CDN 供給業者により行われる．CDN サービス Akamai [54] は，DNS の機能を置換してキャッシュを持つサーバに置き換える名前解決機能を持つ．CDN サーバを視聴者に近いエッジ部分に配置すると通信経路長を大幅に短縮でき，遅延時間も安定する．文献 [55] では安定した遅延時間の報告がある．コンテンツの人気を考慮した協調型キャッシュ方式での CDN では，平均経路長 5.4 の経路上で全てキャッシュした場合 5.0 に，確率を考慮した場合 4.2 に低減した報告がある [56]. CDN で平均経路長を短くするには，端末側のキャッシュ容量を大きくする必要

があり経済面の問題がある．文献 [57] では，CDN サービスのダウンロード時間を 2000 年と 2001 年で比較している．2001 年では多くの会社でダウンロード速度を改善しているが，日々コンテンツは増加している．特に 2005 年以降では Web2.0 の普及により CGM コンテンツが急増した．ダウンロード時間を改善するためには，各サービス供給者で速度を維持するための計測と CDN サーバの配置変更およびコンテンツ保持容量を調整し続けなければならないため自動的な最適化機能が求められる．

2.4 FIA（Future Internet Architecture）

従来の TCP/IP アーキテクチャでは，多様な利用に対して制約があるため，FIA（Future Internet Architecture）が研究されている．

CCN（Content-Centric Networking）[58, 59] では，コンテンツを中心とした通信基盤を検討している．DTN（Delay-Tolerant Networking）[60] や CNF（Cache aNd Forward）[61] は，接続が不安定な場合を考慮している．DTN，CNF および Breadcrumbs [62] は，通信経路上のルータに名前解決と経路選択の機構を持つ．DTN，CNF はデータの送信側から，Breadcrumbs は受信側から名前解決を行っている．

2.4.1 DTN（Delay/Disruption Torelant Network）

DTN (Delay/Disruption Torelant Network）[60] では，通信経路上の各リンクが常時接続できない環境に対応して接続機会を増やすアーキテクチャが提案されている．中継ノードは 最終受信ノードになるべく近くなるようデータを転送する．各中継ノードにルーティング機構とキャッシュ機構を持つ．

DTN では，複数の転送先にデータを転送する．転送先を増やすことにより送信できる可能性が高められるが，不用意に転送先を増やすと利用されないキャッシュ容量を消費し経済面で不利となる．文献 [63] では，移動する端末の通信に対して，転送先に冗長性を持たせることでキャッシュヒット率を向上させている．文献 [64] では，リンク間の利用率類似度とリンク間の利用率で送信先を選定するルーティング方式により遅延時間を 2/3 に，平均経路長を 1/4 に改善した結果を提示している．

DTN の要求方向について検討した文献 [65] では，プッシュ，プル，複製の機能を検討している．動画視聴においてもコンテンツ供給者側からのプッシュ機能とコンテンツ視聴側からのプル機能

およびコンテンツ供給者とコンテンツ視聴者が意識せずに通信経路上で複製を行う機能が考えられる．

DTN の輻輳制御ポリシ [66] では，ストレージルーティングが提案され，メッセージ選択，ノード選択，取得選択の順に処理する方法が述べられている．その中で，ポリシは，pushTail, pushHead, pushOldestNetworkAge, PushLatestRouteAvailability, pushSmallest, pushLargest, pushLowestPriority が紹介されており，動画視聴のキャッシュ処理でも複数のポリシから選択する方法が考えられる．

2.4.2 CCN（Content-Centric Networking）

CCN は伝統的な機能アプローチの問題点からコンテンツを中心としたアプローチのアーキテクチャを検討している．CCN フレームワーク [58] では，コンテンツを中心とした新たなフレームワークを提示している．CCN では，コンテンツ，サービス，利用者の権限に応じて適した利用条件に合うコンテンツを利用者に提示する．CCN ではキャッシュ情報を共有しない粗結合のキャッシュルータが想定されている．

各種アプリケーションに向けた実装も検討されており，VoCCN [67] では，CCN 上での電話機能を対象としている．

しかし，CCN は，総合的なアプリケーションに対応するフレームワークを指向しているため，名前解決やデータ取得方法の具体的検討が不足している．

2.4.3 CNF（Cache-aNd-Forward）

CNF では，DTN のように接続性が保証されないネットワークにおいて，受信したデータをキャッシュして転送先を選定し，送信する機能を中心に検討している．文献 [68] では，名前解決，モバイル端末を含めたプロトコルシーケンスが提示され，その評価では負荷が増えた場合に TCP プロトコルより性能が良くなることが示されている．

CNF は，送信者・受信者が明確であり，次のリンクへデータを送信できれば，データをキャッシュから廃棄する方法が多い．このため不特定多数の視聴者が長期間アクセスする可能性が高い動画視聴では，キャッシュ容量が膨大となるため向いていない．

2.4.4 NDN（Named Data Network）

名前解決の新たな枠組みとして NDN [59] が検討されている．NDN では，コンテンツ情報を Interest Packet として管理している．文献 [69] の評価では，コンテンツ供給者がコンテンツ情報をアナウンスすると購読者へマルチキャストするモデルを用いているが，経路上のキャッシュを増やすと照会数を増加させなくても制作者側の負荷の増加に対応できるが，ネットワーク全体のトラヒックはキャッシュによらず増加する．コンテンツ情報の宣伝の方法について不用意に拡散しない方法が望まれる．

NDN を用いたアドホック接続の評価 [70] では，移動端末の割合が 25%を超えるとデータ転送量が 1 /3 以下になる可能性があることが示されている．動画視聴に向けたサービスでは視聴者がどこにいても適切な視聴ができるネットワークが望まれる．

2.5 コンテンツダウンロードの課題

通信経路上の各ノードで QoS（Quality of Service）を確保する SLA（Service Level Agreement）を設定したモデルの評価 [71] では，単位時間あたりのセッション要求数が増えると成功率が低下し，最悪ケースでは 5%以下となる報告がある．動画ダウンロードにおいて人気の高いコンテンツの出現はコンテンツ配信基盤にとって対応が必要と考える．

TCR 以外の配信基盤は，いずれも動画視聴で求められるタグ検索に未対応である．

CCN の文献 [58,72] では，コンテンツを適切な場所から取得するために粗結合のキャッシュルータを仮定しているが，動画視聴の課題検討が不十分である．文献 [59] では，コンテンツ情報を Interest Packet として，コンテンツデータを Data Packet として管理し，ルーティングやセキュリティを考慮しているが，派生コンテンツが多数制作される CGM や SNS によるアクセスランキングの動的変化に対応して，より簡便な方法が望まれる．

DTN や CNF では，経路上のルータが可能な限り受信者側へデータを蓄積転送するが，視聴者側からの要求による駆動ではない．

Breadcrumbs では視聴者側からの要求により名前解決機構が駆動される．性能を改善した Hop-aware BC [73] が提案されている．共に機構のキーは URI であるが，キーをタグにすればストレージコストを改善できる可能性がある．

経路上でコンテンツをキャッシュするコンテンツ配信基盤の性能評価は，CMS をネットワーク中央に配置した Content-Centric Router [74] や，格子状ネットワークでの TCR [4] があるが，より現実的なネットワーク環境での評価が求められる．

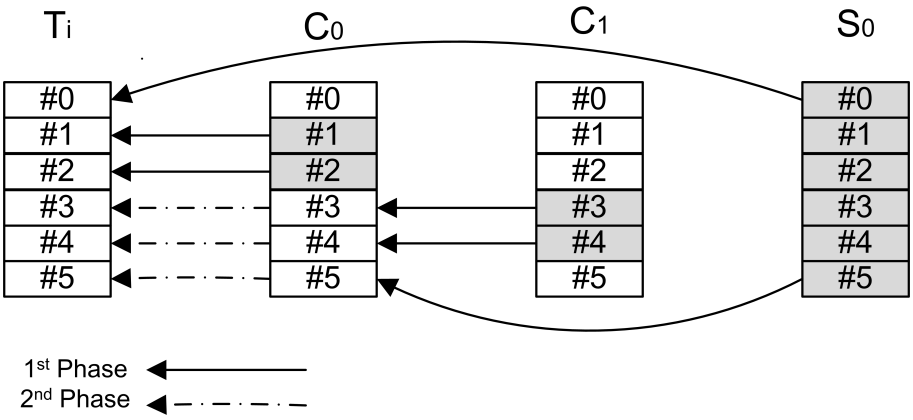


図 2.1: 階層キャッシュによる断片化データの取得

2.6 コンテンツ配信基盤の要素技術

2.6.1 データ複製

複数の複製先を持つ場合には，スケーラビリティが重要となる．複製プロトコルの比較をした文献 [75] では，Read-One,Write-All と Quorums(Majority, Tree, Grid) の比較を行っているが Grid は，Majority に比べノード数の増加に対する可用性が低い．

Google File System [76] では，データが更新されないことを利用して複数の複製先にデータを送信する．全て送信先がデータを更新しなくても次のデータを更新することでデータの複製速度を高めている．

階層キャッシュによるコンテンツデータ取得を図 2.1 の例を用いて説明する．CMS ノード S_0 にあるコンテンツは，一定の固定長に分割した際，#0 から#5 までの 6 ブロックに分割されたとする．ここで灰色で塗られているブロックは存在することを意味する．端末 T_i は，ノード構成として，C_0，C_1，S_0 の経路リストを構成したとする．初期にキャッシュの存在確認をしてブロック#0 は，ノード S_0 が最寄りの存在ノードであるため，S_0 からブロックを取得する．ブロック#1，#2 は，ノード C_0 に存在するため，C_0 から取得する．ここで，C_0 が C_1 にブロック#3 および#4 が存在していることから C_0 に，同様にブロック#5 を S_0 より取得できたとする．次に T_i は，ブロック#3 以降の存在確認をした際に C_0 より取得すればよい．

一般的に C_0 を ISP (Internet Service Provider) 内に，C_1 を C_0 より上位接続とすると，遅延時間 D は，$D_{T_i toC_0} < D_{T_i toC_1} < D_{itoS_0}$，回線速度 V は，$V_{T_i toC0} \leq V_{C0toC_1} \simeq V_{C_1 toS_0}$ なので，端末側にコンテンツデータを置くことで高速化が図れる．キャッシュのヒット率が 0 であった場合でも，キャッシュサーバが存在しない現状の通信に比べて，若干の制御パケットの通信が発生するのみ

表 2.1: 階層キャッシュの登録方法

方法		概要
LCE	Leave Copy Everywhere	経路上全てに複製
LCD	Leave Copy Down	ヒットしたノードの下位に複製
MCD	Move Copy Down	ヒットしたノードの下位に移動

表 2.2: キャッシュ廃棄方法

方法		廃棄対象
LRU	Least Recently Used	アスセスが最も古いもの
LFU	Least Frequentry Used	一定時間内でアクセスが少ないもの
FIFO	First-In, Forst-Out	登録順
random	random probablity	確率により無作為に選択

であり，ダウンロード品質の悪化への影響は少ない．

2.6.2 キャッシュ置換方法

データを断片（チャンク）として分割してキャッシュし，検索する方法が提案されている．分散環境でのオブジェクト管理に Key Value Store があり URI のようなコンテンツを特定する ID にブロック番号を加えてキーとすることで分散環境でのコンテンツのデータ断片の管理が可能となる [77]．しかし，どの部位をどのノードで保持しているか名前解解決を行う機能が必要となる．静的コンテンツの場合は，コンテンツの寿命は長いため，DNS（Domain Name System）のように場所を登録するサーバを固定しても良い．CGM コンテンツの場合は，コンテンツ寿命が不定であるため名前解決を行うサーバの位置を決めてしまうと，そのサーバとの情報交換トラヒックが多発すると考えられるため視聴端末からオリジナルコンテンツを持つ CMS への通信経路上のいずれかのノードで名前解決を行う方法が求められる．

表 2.1 にキャッシュ登録方法を，表 2.2 にキャッシュ廃棄方法を示す．

階層キャッシュ構成方法として，データ保持ノードから取得要求発行ノードまでの経路上全てのノードにキャッシュする LCE（Leave Copy Everywhere），ヒットしたノードの 1 段端末よりに複製する LCD（Leave Copy Down），ヒットしたノードの 1 段端末よりに移動する MCD（Move Copy Down）などがある [78]．キャッシュ置換をコストにより変更する方法 [79] があるが，経路上のノードの状態を把握するための経路上のノード間の通信が必要となる．確率による廃棄と LCD による

廃棄による経路短縮をシミュレーションした文献では，トラヒックが多くなると確率による廃棄よりLCDの方がキャッシュヒットが高くなる傾向が報告されている [80]．Zipfアクセス分布によるキャッシュ置換方法の評価ではLCDの方が遅延時間は少ないが，より現実的なトラヒックを用いた場合は，データ量の1.5%のキャッシュサイズにおいて，LCEとLCDは同等となっている [81]．経路長が10のネットワークでのキャッシュ登録方式をLCE, LCD, またはMCDで，キャッシュ廃棄方式をLRUまたはFIFOで比較したシミュレーションでは，いずれの場合もキャッシュヒット率が65%程度の結果が提示されている [82]．

以上から，人気がSNSにより急変する場合のキャッシュ方式は，他のノードの影響によらず単純な方法が効果を得られると考える．

2.6.3 ネットワークスケーラビリティ

センサーネットワークでは，ツリーの葉にあたるセンサーノードのデータを無線により枝，幹にあたるセンサーノードに送信し，最終的に根にあたるノードでデータを収集する．センサーネットワークでは，ノード数が多いほど消費電力が少ないという報告がある [83]．CMSにあるコンテンツデータを視聴端末で取得する方法と方向が逆となるが，データをキャッシュするノードがある程度あれば，ツリー状のコンテンツ配信経路でも平均通信経路長が削減可能と考えられる．

2.6.4 ランキング分布

Web2.0でのアクセス分布はロングテール状となることを前述したが，CGM動画コンテンツの時間的なアクセス数の変動は以下の3種類が想定できる．

ブーム型　人気の有るコンテンツの情報がSNSにより急速に伝搬し，アクセスが急上昇し，一定時間経過するとアクセスはほとんど無くなる．

ロングテール型　評判が緩やかに伝搬したり再視聴が多いことから，アクセスは少なくなるが長時間にわたりアクセスが継続する．

コンスタント型　アクセスは多くないが長期間に渡り一定のアクセスがある．

本研究では，長期間のコンテンツアクセス数の変動の影響は，保留して，1時間程度にアクセスされるタグのアクセス数の分布に着目し配信基盤の評価を行う．

ロングテールの近似として冪乗則（式 2.5），パレート分布（式 2.6）[84] があり，パレート分布の離散型として表す Zipf 分布 (式 2.6）がある．

$$\text{冪乗則} \quad f(x) = ax^k \tag{2.4}$$

$$\text{パレート分布 (累積)} \quad F(x) = 1-(k/x)^\alpha \qquad where\ \alpha, k > 0, x \leq k \tag{2.5}$$

$$Zipf\ \text{分布} \quad f(\kappa; s, N) = \frac{1/\kappa^s}{\sum_{n=1}^{N} \frac{1}{n^s}} \qquad where\ s = 1 \tag{2.6}$$

式 2.6 の N は全要素数で，s が 1 のときに一般の zipf 関数となる．しかし，要素数が少ない場合に扱い易くするため，本研究では，式 2.7 を導入する．スケールフリー性からアクセス頻度の少ないタグがロングテール状に存在する．割引率 d は，ロングテールの程度を表す．K は対象とするタグの要素数で，k はタグの順位である．

式 2.7 の $\sum_{n=1}^{k} \frac{1}{n}$ 部分は，Zipf 関数である．割引率 d はランキング第 1 位の生起確率を 1 とした比であり，K 位から N 位までのランクは d の生起頻度となる．d が大きいほど上位ランクのアクセス頻度が少なくなる．ロングテール部分の長さを全タグ数 N で，生起確率を d で実際の分布と比較しやすくする．これにより，K 位から N 位までは同一の生起確率となる．

$$D_{k;d,K} \equiv \begin{cases} (1-d)\left(\sum_{n=1}^{k} \frac{1}{n} - \sum_{n=1}^{K} \frac{1}{n}\right) + d & (1 \leq k < K) \\ d & (K \leq k \leq N) \\ 0 & (otherwise) \end{cases} \tag{2.7}$$

2.6.5 キューイングモデル

リトルの公式（Little's formula）[85] より W を平均待ち時間，L を平均キュー長，λ を平均到着率とすると式 2.8 となる．

$$W = L/\lambda \tag{2.8}$$

ケンドール記号 M/M/1/∞/∞/FIFO の場合（到着がポワソン過程，サービス時間が指数分布に従う，窓口数 1，容量 ∞，来客数 ∞，先着順サービス）の場合，平均サービス率を μ，窓口利用率

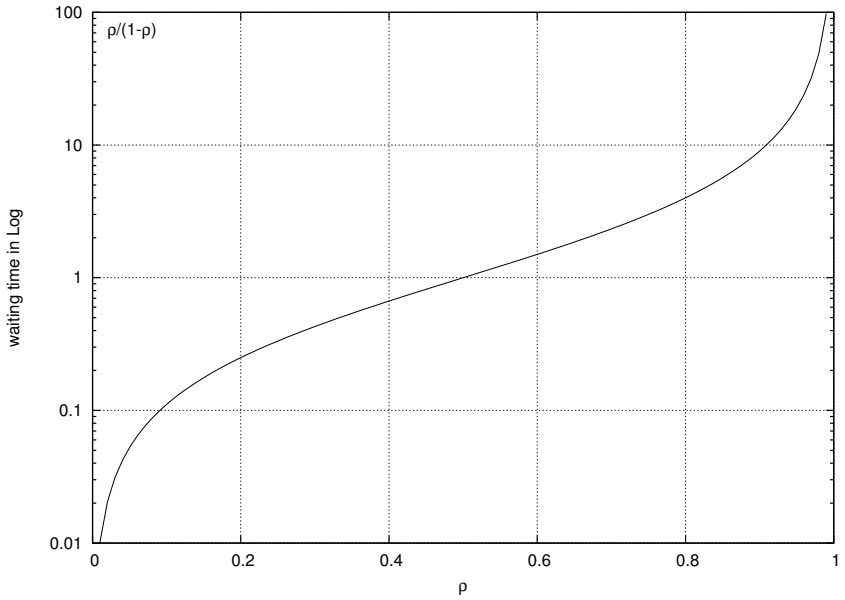


図 2.2: M/M/1 モデルの平均待ち行列長

を ρ とすると平均待ち時間は，式 2.10 となる．

$$\text{平均待ち時間}\ T_q = (1/\mu)/(1-\rho) = T_s/(1-\rho) \tag{2.9}$$

$$\text{待ち行列の平均長さ}\ L_q = \rho/(1-\rho) \tag{2.10}$$

図 2.2 に M/M/1 モデルの平均長さを示す．待ち行列理論より平均待ち時間は $\frac{T_s}{1-\rho}$, 平均行列長は $\frac{\rho}{1-\rho}$ のように回線負荷 ρ が増えると急増するため，速度 V_i を上げるには，速度が大きなノードのヒット率をあげ，CMS 側回線の負荷を軽減することが有効である．M/M/1 モデルでの待ち時間は，窓口利用率 ρ に応じて図 2.2 のようになる．利用率 60%で，2.3 倍，80%で 4 倍，100%で 20 倍の待ち時間となる．利用率をボトルネックとなるリンクの回線負荷と考えると，回線負荷が 80%から 50%になると待ち時間は 1/4 に，60%から 50%に成った場合でも 1/2 に削減できる．回線負荷を低減することが，ダウンロード時間の削減に結びつく．

図 2.3 にキューイングモデルを示す．各 TCR では，$link_n$ より受信したパケットを出力先のリンク $link_m$ の待ち行列 $queue_m$ に入れ (p_{m0})，出力先の待ち行列が空いていれば，$xmitter_m$ で (送信データ長/回線速度) に応じて待ち処理を行う簡単なキューングモデルで処理することを考える．

入力したデータは，自身へのキャッシュ判断を行うため cachingFilter() で前処理を行う．

このモデルでは，要求に対する応答を返す場合は，出力 $link_m$ はノード n 向きとなるので，両端のノード n, m 間のリンクで $n \rightarrow m$ と逆向きの $n \leftarrow m$ の 2 つのリンクを作成して出力側に待

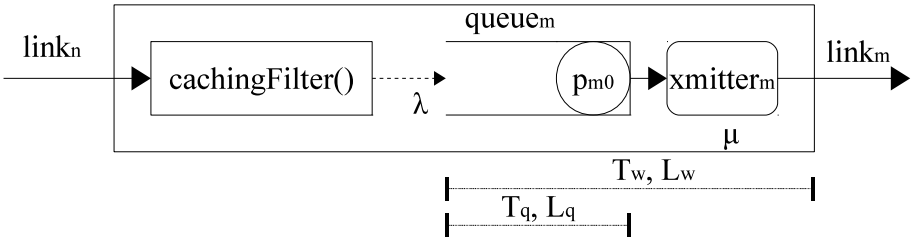


図 2.3: キューイングモデル

ち行列を用意すればよい．

ここで，平均待ち行列時間 T_w，待ち行列の平均長さ L_w，平均待ち時間 T_q，および待ち行列の平均長さ L_q は，多段のキャッシュ構成の場合に論理的に予測することは困難であるためシミュレーションにて評価する．

TCR アーキテクチャの概略モデルを図 2.4 に示す．

複数の視聴端末 $Viwer$ はエッジルータである TCR_1 に接続される．各 TCR はルータ機能 R とキャッシュ C を有する．CMS はルータ機能 R とオリジナルコンテンツを保持するストレージ S からなる．TCR 間のリンクは，CCN で見たようにキャッシュ情報を共有しない粗結合で状況に応じて適切な経路を各 TCR が選択する階層キャッシュ構成をとる．

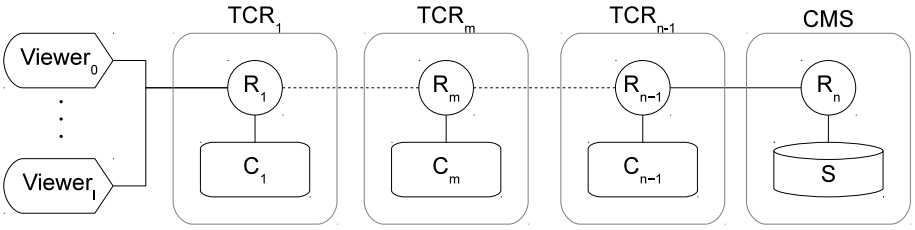
端末側からキャッシュは，$C_0, C_1, \cdots, C_{m-1}$ と並び，オリジナルコンテンツは，S_0 にあるときにコンテンツのダウンロード時間 t は 取得ノード $_i$ の速度 V_i，ヒット率 H_i とすると，式 2.11 で求めることができる．

$$t = \mathsf{sizeOf}(Content) / \sum_{i=C_0}^{C_{m-1},S} V_i \cdot H_i \tag{2.11}$$

2.7 まとめ

フォークソノミーを用いたランキング集計を CMS で集中的に行うと膨大な計算資源が必要であり，検索精度を高めるためには集計頻度を高める必要がある．しかし，アクセス分布がロングテール状で，アクセス数の今後の予測が困難な CGM 動画には，CMS サイトの集計は経済的にデメリットとなる．

既存のコンテンツ配信基盤では，ネットワークトラヒック増加の対応を行っているが，新たな利用ニーズに対応しにくいため FIA が検討されている．いずれの方法でも動画検索で多用されているタグによる検索に対応していない．



TCR: Tag Caching Router, CMS: Content Management System,
R: Routing function, C: Caching function and Data, S: Storage

図 2.4: キャッシングルータモデル

CGM コンテンツの動画視聴の特性を利用してタグアクセス分布の上位のコンテンツ情報およびデータを通信経路上にキャッシュすることで，経済面，性能面で優れたコンテンツ配信基盤を提供することができる可能性が高い．

（意図的な空白ページ）

第3章　TCR（Tag Caching Router）アーキテクチャ

タグ検索とコンテンツダウンロードの両方を効率的に支援する TCR（Tag Caching Router）アーキテクチャについて述べる．

3.1 節で TCR アーキテクチャの概要を確認し，他のコンテンツ配信基盤との相違を明らかにする．3.2 節で TCR と外部機器の関係を明らかにし，3.3 節で動作シーケンスを，3.4 節で TCR のアルゴリズムを述べる．最後に 3.5 節で TCR アーキテクチャについてまとめる．

3.1　TCR（Tag Caching Router）アーキテクチャの概要

3.1.1 節で TCR アーキテクチャの目的を，3.1.2 節で TCR アーキテクチャの機能を，3.1.3 節で従来方法との相違を，最後に 3.1.4 節で外部構成について述べる．

3.1.1　TCR（Tag Caching Router）アーキテクチャの目的

TCR アーキテクチャの目的は，以下の 3 点である．

1. 視聴者が視聴候補コンテンツリストをタグにより容易に取得できる
2. ダウンロードが集中しても性能の悪化を抑制する
3. 推奨コンテンツを宣伝し視聴者の視聴機会を増加する

タグによるコンテンツ検索：　音楽視聴では，ジャンル，タイトルまたは演奏者など検索手法が確率しているため，CMS 側の提供する検索機能を視聴者が利用できる．動画視聴では，商用動画は音楽と同様に検索が可能であるが，CGM では，検索手法が確立しておらずコンテンツ供給者，視聴者がコンテンツの特徴を表すタグをコンテンツに付けてフォークソノミーにより分類する．コンテンツに割り当てられたタグは，流動的で，視聴者により適切なものに置き換えらる．視聴者は，自分の関心のあるタグを含むコンテンツから視聴を決定すること

で，視聴者満足度が向上する．以上から，CGM 動画視聴では，タグによるコンテンツ検索が必須である．

ダウンロード時間悪化抑制： 単位時間あたりのデータ量は，動画ファイルは音楽ファイルの数十〜数百倍以上大きく，かつコンテンツの再生時間も 10 倍以上ある．人気の高いコンテンツが CMS に投稿されると視聴が増え，ダウンロードトラヒックの増加によりネットワークの負荷が高くなる．動画視聴では，ダウンロード速度が再生速度より低速になると視聴が中断する恐れがある．動画ダウンロードが増えてもダウンロード時間が悪化しにくい方法が求められる．

推奨コンテンツの宣伝： CMS では，再生数の多い動画をランキング表示し，視聴者の検索精度向上と CMS サイト利用回数増加を行っている．現在では SNS の影響が大きく，人気動画の情報が多くの視聴者へ瞬時に伝達されるため，ネットワーク，CMS サーバの負荷集中が発生する．負荷集中が発生すると多くの視聴者に視聴中断が発生するため，緩和する方法が求められる．

3.1.2 機能

上記の目的を達成するために必要となる機能は以下である．

1. タグによるコンテンツ検索
2. 検索・ダウンロード経路の探索と最適化
3. 経路上へのキャッシュ
4. 推奨コンテンツの拡散

各機能について説明する．

タグによるコンテンツ検索： 視聴者がタグを指定し検索するとそのタグが割り当てられた視聴候補コンテンツリストを返す．検索結果は，経路上で得ることができれば，オリジナルコンテンツを保持する CMS まで要求は到達しなくても応答は視聴端末に戻すことができる．

検索・ダウンロード経路の探索と最適化： 経路上のルータは，視聴端末から発行された検索要求またはダウンロード要求に応じて情報取得経路を探索し，状況に応じて経路を改善する．

経路上へのキャッシュ： 経路上のルータは，通過するパケットから各ルータの判断によりコンテンツ情報，コンテンツデータまたは照会先経路をキャッシュすることができる．このキャッシュにより通信経路を短縮する．

推奨コンテンツの効果的な拡散： CMS または SNS は，推奨コンテンツリストを複数のルータに不定期に送信する．これらを受信した経路上の各ルータは，タグに応じて適した経路を選択して送信することで拡散し，視聴端末は推奨コンテンツをアクセスする機会を増すことが可能となる．

キャッシュヒット率を高める方法としてプリフェッチがある．TCR アーキテクチャでは，プリフェッチ機能を持たない．理由は，人気 CGM コンテンツのアクセス変動は激しく，コンテンツ登録後にタグが動的に変更されるため各 TCR ルータでのアクセス予測は困難である．そのため，TCR アーキテクチャでは，プリフェッチの代わりにコンテンツ供給者側からの推奨コンテンツのアドバタイズの機能により，キャッシュヒット率向上の手段を導入している．

TCR アーキテクチャを採用するメリットについて記す．

CDN では有償コンテンツ視聴者や速度重視の法人向けに有償サービスを提供することでストレージコストを回収できるが， CGM コンテンツは無償または低価格での提供サービスであるためストレージコストを確保することは難しい．ISP がコンテンツプロバイダと提携して有償コンテンツを加入者にサービス提供しているが購入者は少ない．TCR アーキテクチャを導入した ISP は，少ないストレージコストの投入で人気コンテンツの視聴機会が増すので低価格競争を行っている ISP には付加価値とすることができる．100 ノードの BA(Barabási-Albert) [86] ネットワーク，データサイズ 100MB，コンテンツ数 100 の環境では CDN の場合に約 60 ノード ×100MB×100 コンテンツのストレージが必要になるが，TCR ではタグランキングのアクセス分布が良好（割引率 d が小さい）であれば，100 ノード ×100MB×4 コンテンツ程度のストレージ容量で同等のダウンロード時間が実現できる．実際のコンテンツ数はニコニコ動画で 900 万程度なので，上位 1 割を対象とした場合でも CDN では多大なストレージコストを要する．

3.1.3 従来方法との相違

インターネット上に存在するサーバからデータを取得する方法は，インターネットの利用が進展するにつれ改良されてきた．表 3.1 に配信基盤の比較を示す．視聴者は，検索エンジン SE，SNS または口コミ等で得たコンテンツの情報を元に CMS よりコンテンツを取得する．

伝統的な TCP/IP の FTP や HTTP では，コンテンツを特定する FQDN から DNS により CMS サーバの IP アドレスが取得できるため，IP ルーティングによりデータ取得要求は順次転送され，CMS サーバからは応答が返される．

CDN では，DNS の名前解決機構を変更して視聴者の近くのエッジサーバにキャッシュされている IP アドレスを返すことで通信経路の短縮が図ることができる．

表 3.1: 配信基盤の比較

大項目	小項目	TCR	Traditional	CDN	P2P	CNF
検索	キー	Tag	URI	URI	Hash key	GFID
	関係	n:m（非可逆）	1:1	1:1···n	1:1···n	1:1···n
レゾルバ	型	キャッシュ表	DNS	DNS 置換	Tracker	キャッシュ表
	入力	Tag	URI	URI	Key	GFID
	出力	URI	IP	IP	node	next hop
	重複	有り	有り	有り	有り	無し
	場所	経路上	エッジ	エッジ	ホスト	経路上
キャッシュ	場所	経路上	N/A	エッジ	利用者	経路上
	寿命	短期間	N/A	恒久的	使用数	受信完
	契機	受信	N/A	供給者	取得	受信

P2P では複数のユーザ側の PC にコンテンツデータの断片を取得状況によりダウンロード中にキャッシュすることで，サーバへの負荷集中を回避し，同時期に利用する利用者が多いほどダウンロード性能が改善される．

CCN の一技術である CNF では蓄積転送することで受信側の受信機会を増すことを志向している．

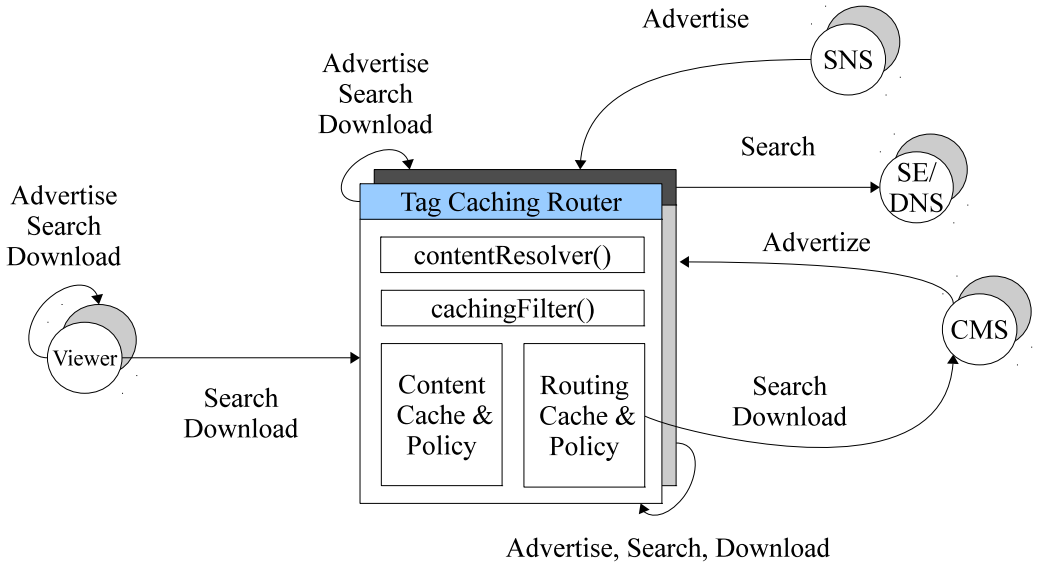
TCR では，タグをキーとしていることが他の配信基盤と異なる．従来の配信基盤では，コンテンツを一意に特定するためのキーを指定する．アプリケーションプログラムがコンテンツを利用する場合には指定したコンテンツと得られたコンテンツデータが異なると不都合が起きるが，利用者が人間である場合にコンテンツを一意に特定する必要がない．

3.1.4 外部構成

TCR は，IP ネットワーク上で動作するアプリケーション層のルータで，ネットワーク上に複数配置される．TCR は，視聴端末 Viewer，オリジナルコンテンツを保持する CMS，コンテンツ情報を取得できない場合の照会先である SE（検索エンジン）／ DNS（Domain Name System），推奨コンテンツを宣伝する CMS と SNS，および他 TCR と連動して動作する．

SNS，CMS は推奨する複数のコンテンツ情報を格納したアドバタイズ要求を，事前に設定した TCR に送信する．Viewer は，視聴者がタグを指定して視聴コンテンツを探索した時点でコンテンツ検索要求を，視聴コンテンツ決定時にダウンロード要求を事前に設定した既定の TCR に送信する．図 3.1 に TCR と他機器の関連を示す．

TCR： 各 TCR は，パケットを受信すると，パケットと内部に保持しているキャッシュに応じて処理を行う．コンテンツ検索要求またはダウンロード要求を視聴端末 Viewer または他 TCR



SNS: Social Networking Service, SE: Search Engine, CMS: Content Management System

図 3.1: TCR：外部構成

から受信すると内部のキャッシュを調べ，応答を返せる場合は，パケットを生成して応答し，キャッシュが存在しない場合は，転送先を選定して要求を転送する．また，アドバタイズ要求を受信した場合は，要求に含まれる推奨コンテンツリストのタグに応じて転送先を決定し，アドバタイズ要求を拡散する．各 TCR は，アドバタイズ要求，検索応答，ダウンロード応答を受信した際に各 TCR のキャッシングポリシに応じて内部に受信したパケットのデータを保持するか判断する．

SNS： 各 SNS は，不定期にコンテンツのランキング情報や推奨コンテンツリストを既知の TCR にアドバタイズ要求として送信する．

CMS： 各 CMS は視聴者またはコンテンツプロバイダから投稿されたコンテンツを保持し，視聴者にコンテンツの投稿，検索，視聴機能を提供するとともに視聴者のコンテンツ選定を支援するために視聴ランキング提示，タグ編集機能も提供する．また，不定期に SNS と同様に推奨コンテンツリストを既知の TCR を提供する．TCR からコンテンツ検索要求またはダウンロード要求を受信した場合には，要求に応じた応答を返す．

Viewer： コンテンツ視聴者は，視聴端末 Viewer に搭載されたブラウザのような視聴アプリケーションプログラムから，自分の関心のあるコンテンツの検索を行なうと既定の TCR にコンテンツ検索要求が送信され，その TCR から視聴候補コンテンツリストを得ることができる．

得られたコンテンツリストより視聴するコンテンツを選択すると既定の TCR にダウンロード要求が送信され，その TCR からコンテンツデータの固定長の断片を取得できる．Viewer はダウンロード要求の断片番号を順次増加させ，コンテンツデータが終了するまでデータを順次取得する．Viewer は取得したコンテンツデータの断片を逐次ブラウザに渡し，視聴と並行して Viewer でのデータダウンロードが行われる．視聴者は，CMS の提供するタグ編集機能を各自の判断に応じて適宜行う．

SE/DNS： TCR は他の TCR から要求された要求の転送先が見つけられない場合は，最後の手段として，SE と DNS を使用して推奨コンテンツリストまたはコンテンツデータの取得を行う．

3.2 TCR（Tag Caching Router）の構成

図 3.2 に TCR の構成を示す．

TCR はパケット受信時に外部構成 3.1.4 節で述べた様に，Viewer，CMS，SNS，SE/DNS，または他の TCR と通信を行う．

TCR は，コンテンツ関連キャッシュ ContentTable，探索経路キャッシュ RoutingTable，経路探索機能 ContentResolver() およびキャッシュ可否判断機能 cachingFilter() で構成され，IP 層の上に実装される．

各 TCR に設定する設定値については 3.2.1 節で，パケット構造については 3.2.2 節で，キャッシュデータの詳細は 3.2.3 節で説明する．

各 TCR は，アドバタイズ要求，コンテンツ検索要求の応答またはダウンロード要求の応答を受信した際に cachingFilter() がキャッシュ管理を行う．また，各 TCR は，コンテンツ検索要求またはダウンロード要求を受信した際に自身のキャッシュから転送先ルータの選択を contentResolver() が行う．

3.2.1 設定値

以下にネットワーク管理者が各 TCR に事前設定して TCR の動作を制御するためのパラメータを示す．

既定ルータ： 転送先ルータが見つからない場合に要求を転送するルータであり，自身に対して近傍の TCR を RouteByTag に 1 つ設定する．

キャッシュ登録方式： 通信経路上の一連のキャッシュに登録する方式

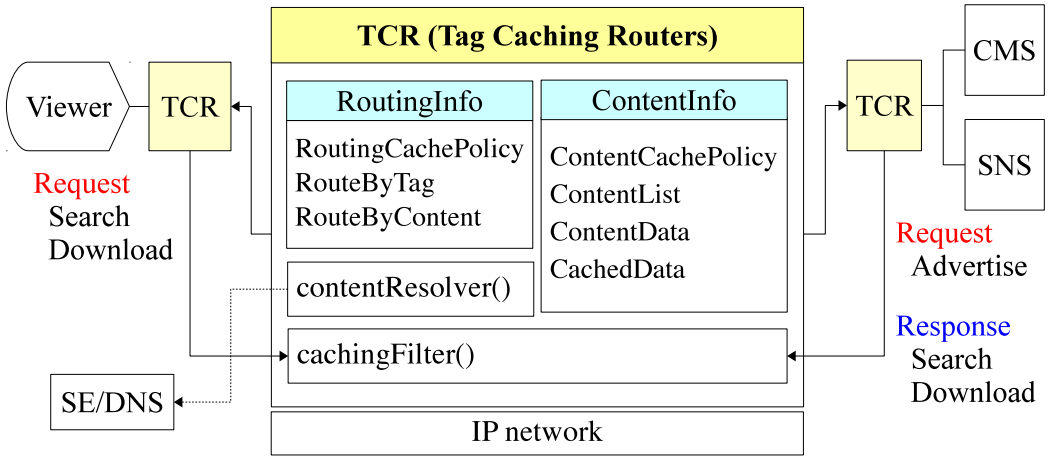


図 3.2: TCR：タグキャッシングルータの構成

キャッシュ廃棄方式： キャッシュエントリ数が上限に達した場合の廃棄する方式

ルーティングキャッシュポリシ： パケットの受信により知り得た転送先ルータをキャッシュする基準

コンテンツキャッシュポリシ： パケットの受信により知り得たコンテンツ情報やコンテンツデータをキャッシュする基準

キャッシュの登録・廃棄方式は，階層キャッシュで用いられている方式 [78] より選択する．'|' は，左右いずれかの選択を表す．

```
EntryMethod    := LCE|LCD|MCD
DiscardMethod := LRU|FIFO
```

キャッシュポリシの書式は Apache [87] のアクセス設定と同様で，`access` は許可（allow）または禁止（deny）を指定する．

RoutingCachePolicy := <access, *Tag*, Router>

ContentCachePolicy := <access, *Tag*>

ここで，`Router` は，転送先ルータの IP アドレスである．キーは斜体で表している．

3.2.2 パケットデータの構造

TCR では，アプリケーション層に，要求および応答パケットを実装する．パケットの種類は，アドバタイズ要求 `AdvReq`，コンテンツ検索要求 `SearchReq` とその応答 `SearchRsp`，コンテ

ンツダウンロード要求 DownloadReq とその応答 DownloadRsp の 5 種類である．アドバタイズ要求は視聴候補のコンテンツリストを広く拡散することを目的としているため応答は存在しない．それぞれのパケットの構造を示す．

```
AdvReq:      requester, to, cnt, [<Tag, URI, Meta>]
SearchReq:   requester, to, [<Tag|URI>]
SearchRsp:   requester, to, cnt, [<Tag, URI, Meta>]
DonwloadReq: requester, to, [URI,BlkNo]
DownloadRsp: requester, to, cnt, [URI,BlkNo,Data]
```

パケットヘッダは，パケット識別子に続いて要求元 requester，受信対象 to とキャッシュ登録方式用のカウンタ cnt で構成される．不特定ノードに対する要求は，to を'*' として使用する．

cnt は階層キャッシュ構造を制御するためのカウンタで，検索応答またはダウンロード応答を生成する TCR がキャッシュ登録方法 EntryMethod に応じて設定する．検索応答または，ダウンロード応答を受信した TCR は，受信パケットの cnt 値を減じて要求元へ転送する．cnt が負値になるとキャッシュ更新を行わないことで LCD または MCD を実現する．

'[ ]' はペイロードで，パケット識別子により上記のように構造が異なる．'< >' はタプルを表す．AdvReq の Meta はコンテンツの作者，概要，ジャンルなど，視聴者がコンテンツ選択に利用するためのコンテンツの特徴を表すメタ情報である．TCR ではコンテンツデータを固定長の断片でキャッシュする．DownloadReq または DownloadRsp の BlkNo は，コンテンツデータ断片の順序を表す．

3.2.3 キャッシュデータの構造

TCR アーキテクチャで用いるパケットおよびキャッシュデータの構造について説明する．

キャッシュデータの構造について説明する．

ルーティング用キャッシュデータは，タグをキーとしたルーティングテーブル RoutByTag と URI をキーとした RouteByContent の 2 種類である．参照数 ref はキャッシュの該当データが利用されたときに加算され，キャッシュ選択の優先処理に用いる．

RouteByTag := <*Tag*, *Router*, ref>

RouteByContent := <*URI*, *BlkNo*, *Router*, ref>

コンテンツ関連のキャッシュデータは，ContentList，ContentData および CachedData の 3 種で，それぞれ，視聴候補コンテンツリスト，要求パケットの転送先ルータ，キャッシュさ

れたコンテンツデータの取得に用いる．

ContentList := <*Tag*, *URI*, *Router*, meta, ref>

ContentData := <*URI*, *BlkNo*, *Router*, ref>

CachedData := <*URI*, *BlkNo*. ContentData, ref>

3.3 TCR（Tag Caching Router）の動作シーケンス

要求・応答の流れを図 3.3 で確認し，視聴端末の動作に従った全体の流れを図 3.4 に示し，各フェーズの処理について説明する．

図 3.3 に TCR を用いた場合の要求の流れを示す．

視聴端末 $Viewer$ からコンテンツ検索要求を行うと $Viewer$ の findContent() が既定の TCR に要求を送信する．

要求を受信した TCR は，要求に対応する情報を保持していない場合は自身で応答を返せないため，別の TCR を選定して要求を転送する．この転送は，要求に対して応答が返すことができる TCR まで送信される．転送先の TCR が見つからない場合は，オリジナルコンテンツを持つ CMS に要求を転送し結果を得る．図 3.3 では，TCR_1, TCR_2, CMS の順に要求が転送されている．

応答を受信した TCR の cachingFilter() は自身にキャッシュするか判断したのち，応答の要求元に向けて応答を転送する．図 3.3 では，CMS，$TCR_2, TCR_1, Viewer$ の順に要求が転送されている．

このようにして，視聴端末よりのキャッシュにコンテンツ情報が蓄積される．転送先の選定およびキャッシュ判断は各 TCR により行われるため，経路はその時の状況により変る可能性がある．転送先の選定およびキャッシュ判断方法が優れていれば適した経路からコンテンツ情報を取得できる．

視聴端末 $Viewer$ は検索要求の応答から視聴するコンテンツを選択すると $viewer$ の getContent() よりダウンロード要求が発行される．検索要求と同様に処理されるが，検索要求ではキーがタグに対して，ダウンロード要求ではコンテンツを特定するため URI となる．

また $Viewer$ の要求とは別に，CMS 側の pushContentInfo() から推奨コンテンツを宣伝する流れがある．これは，応答は発生しない．図 3.4 では，図 3.3 をフェーズ別の動作が分るようにした動作シーケンスを示し，アドバタイズメント，コンテンツ検索およびコンテンツダウンロードの概略を説明する．TCR の動作アルゴリズムは 3.4 節で説明する．推奨コンテンツを宣伝するアドバタイズメントフェーズ，視聴コンテンツを検索する検索フェーズ，および視聴するコンテンツのダウンロードを行うダウンロードフェーズについて説明し，初期状態からの動作を例示する．

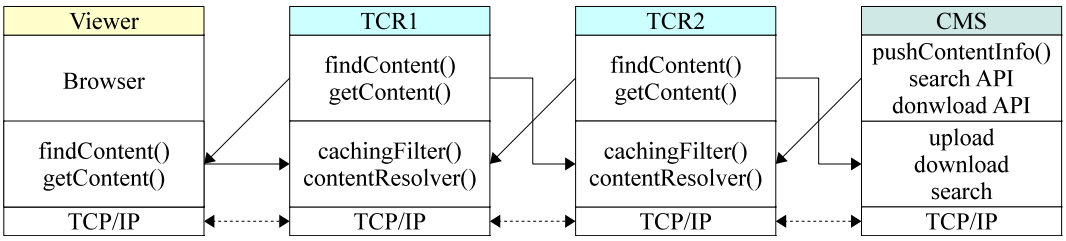


図 3.3: TCR：要求の流れ

3.3.1 アドバタイズメントフェーズ

CMS または SNS は，推奨コンテンツに関するタグ，URI を含んだコンテンツリストを既定の TCR に送信する．アドバタイズ要求を受信した TCR は以下の順で処理を行う．

1. 自身のキャシングポリシによりアドバタイズ要求に含まれる推奨コンテンツ情報をキャッシュデータ ContetList へキャッシュする．
2. アドバタイズ要求に含まれる推奨コンテンツ情報のタグにより転送先ルータを自身の転送先選択用キャッシュ RouteByTag から探す．
3. 該当する転送先ルータごとに推奨コンテンツリストを組み立て直して，アドバタイズ要求を転送する．

キャッシュ処理とパケット転送アルゴリズムの詳細は，3.4 節に示す．

TCR は視聴者側からのコンテンツ検索要求またはダウンロード要求に対する応答によりキャッシュを更新するためアドバタイズ要求を受信しなくても TCR は機能する．

3.3.2 コンテンツ検索フェーズ

視聴者が関心のあるタグを指定して検索すると，Viewer からコンテンツ探索要求が既定 TCR に送信される．この要求を受信した TCR の contentResolver() は，タグが視聴候補用コンテンツキャッシュ ContentList に存在するか調べ，ContentList から選択して視聴候補コンテンツリストを組み立て要求元へ応答を返す．この応答は，タグ，URI などメタ情報が複数含まれたコンテンツリストの形式で返される．ContentList に候補が無ければ，RouteByTag を参照して検索要求を他 TCR へ送信する．転送先ルータが見つからなければ，タグをキーとして SE で検索し，結果からコンテンツリストを生成する．転送先ルータの選択方法は，3.4.2 節で説明し，図 3.9 に示す．

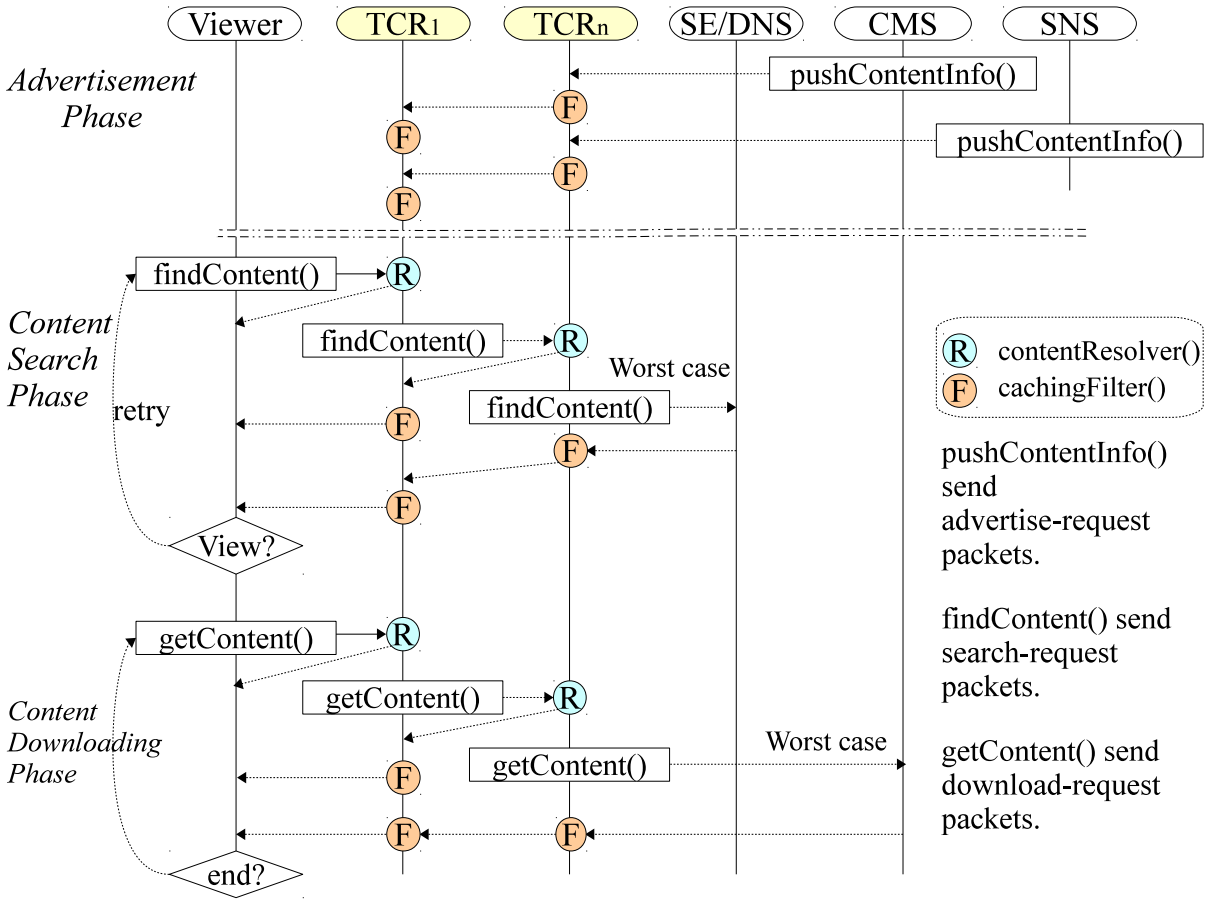


図 3.4: TCR：タグキャッシングルータの動作シーケンス

3.3.3 コンテンツダウンロードフェーズ

視聴者が，受信したコンテンツリストから視聴するコンテンツを決定すると，Viewer は既定の TCR にコンテンツの URI を含んだダウンロード要求を送信する．受信した TCR の contentResolver() は，ContentData からコンテンツデータを保持している TCR を見つけて，要求を転送する．該当 TCR が自身の場合は，CachedData からコンテンツデータを取得して要求元に返す．

このようにして，TCR はキャッシュが空の状態から動作し，動画視聴状況によりキャッシュするコンテンツ情報およびコンテンツを適合させていき，検索およびダウンロードに対して適切な経路を形成できる．

3.3.4 初期動作

図 3.5 にタグキャッシングルータの初期動作を示す．図 3.5 で CMS $S1$ がアドバタイズ要求を発

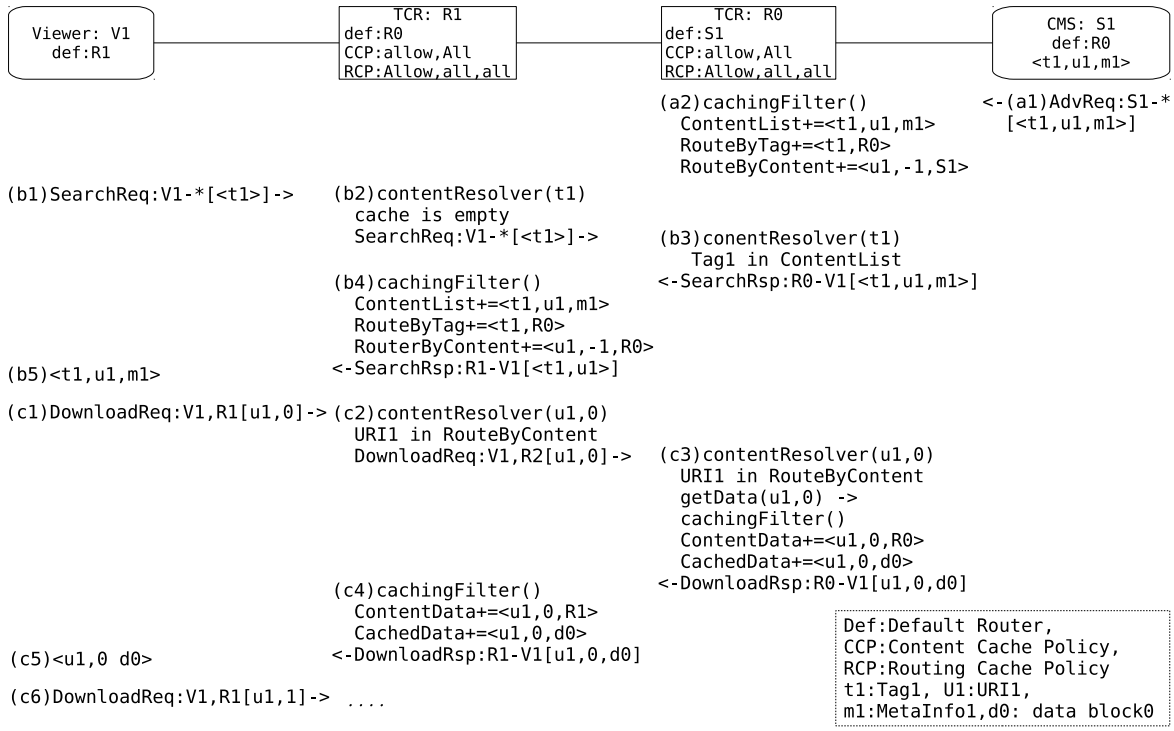


図 3.5: TCR：タグキャッシングルータの初期動作

行するイベントが発生した場合は，以下の動作を行う．

$S1$ は，アドバタイズ要求 `AdvReq<Tag1,URI1><Tag2,URI1>`パケットを既定ルータ TCR $R0$ に送信する．

$R0$ はポリシ CCP が`<Allow,ALL>`のために CachingFilter() が内部に反映する処理を行う．

```
ContentList  += <Tag1,URI1><Tag2,URI2>
RouteByTag   += <Tag1,TCR0><Tag2,TCR0>
ContentByTag += <URI1,0,TCR0>
```

ここで+=は，レコードが存在すれば上書きし，無ければ追加を行うことを意味する．

$R0$ では Tag1 と Tag2 のコンテンツリストを格納したため，RouteByTag，ContentByTag の Router を自分に入れ替えて格納する．

（a1）$S1$ からのアドバタイズ要求ではコンテンツリストは 2 レコードであるが，URI が同一のため，RouteByContent の追加レコードは 1 つとなる．

(a2) 次に，$R0$ は，受信した `AdvReq` をポリシ CRP を参照して転送先を決定する．転送先候補の既知ルータ def は $S1$ であり要求送信元であるため転送しない．RouteByTag は`<Allow,All><Tag1,TCR0><Tag2,TCR0>`であるが，$R0$ は自身であるため転送しない．もし，検索要求やダウンロード要求により RouteByTag にタグに該当する TCR が見つかれば転送を行う．

視聴者が，Viewer $V1$ からタグ Tag1 を検索キーとして視聴コンテンツを検索する．

（b1）$V1$ は検索要求パケット `SearchReq[Tag1]` を既定ルータである $R1$ へ送信する．

（b2）$R1$ は，自信の ContentList を調べ，空のため，既定ルータである $R0$ に検索要求を転送する．

（b3）$R0$ では，ContentList の`<Tag1,URI1>`が該当するため検索応答 `SearchRsp` を要求された $R1$ へ転送する．

（b4）$R1$ の CCP は`<Allow,all>`なのでコンテンツ情報`<t1,u1,m1>`をキャッシュする．RCP も該当するため経路`<t1,R0>`を RouteByTag にキャッシュする．コンテンツデータに関する情報は検索応答には無いためブロック番号-1 を用いて`<u1,-1,R0>`を RouteByContent にキャッシュする．検索応答を $V1$ に向けて転送する．

（b5）視聴端末に検索結果`<t1,u1,m1>`が戻り，視聴者がメタ情報 m1 により視聴を決定すると，URI $u1$ をキーとしてダウンロード要求を開始する（c1）．

ダウンロード要求から応答を得るまでは（b1）から（b5）と同様のシーケンスとなるため省略する．コンテンツデータの終わりまで，$V1$ がブロック番号を増加させてダウンロード要求を発行することでコンテンツデータを取得する．

これにより，CMS $S1$ にあるコンテンツ情報およびデータが，Viewer $V1$ の最寄り TCR $R1$ にキャッシュされたので，以降 $R1$ に接続した Viewer がタグ `t1` またはコンテンツ `u1` をアクセスする際に通信経路長が 1 となり，キャッシュしない場合の 3 から短縮できる．

3.4 TCR（Tag Caching Router）の動作アルゴリズム

3.4.1 節でキャッシュ管理アルゴリズムを，3.4.2 節で転送先ルータの経路選択アルゴリズムを説明する．

3.4.1 キャッシュ管理

TCR では，各 TCR に設定する方式およびポリシに応じてキャッシュ判断を行うため，これらを設定するだけで各種のキャッシュ管理方式を実現でき，加えてタグによるキャッシュ可否を制御できる．各 TCR でのキャッシュ可否判断は，登録方式，ポリシの順に行う．

階層キャッシュ構成方法として，データ保持ノードから取得要求発行ノードまでの経路上全てのノードにキャッシュする LCE（Leave Copy Everywhere）[78] がある．提案する TCR のキャッシュ方式は LCE を想定する．

キャッシュエントリが上限に達した場合の廃棄方式は，従来手法では LRU（Least Recently Used）が広く用いられており，TCR でも既定の廃棄方式は LRU を想定する．

パケット受信時の処理 cachingFilter()

各 TCR がパケットを受信した際に cachingFilter() が駆動される．このアルゴリズムを図 3.6 に示す．以降，紙面の都合より変数またはキャッシュデータに別名を用いる場合がある．その際は，図上部の'as' で定義する．

Input: `recievedPacket as` r

1 **if** type$(r) = (\texttt{AdvReq} \vee \texttt{SearchRsp} \vee \texttt{DownloadRsp})$ **then** $r.cnt \leftarrow r.cnt - 1$
2 $key \leftarrow$ getkey(r)
3 cacheFoward(r, key)
4 **if** type$(r) = (\texttt{AdvReq} \vee \texttt{SearchRsp} \vee \texttt{DownloadRsp}) \wedge \texttt{EntryMethod} = \texttt{MCD}$ **then**
5 removeCache(key)

図 3.6: パケット受信の処理 cachingFilter($recievedPacket$)

キャッシュ登録と転送処理 cacheForward()

図 3.6 の行 2 では，キャッシュ登録方式の処理のためで受信パケットのカウンタ $r.cnt$ を減じ，行 3 でキャッシュ処理とパケット転送を行う関数 cacheForward() を実行する．キャッシュ登録方式 EntryMethod（3.2.1 節）が MCD（Move Copy Down）の場合に自身のキャッシュから該当するエントリを削除するため行 5 で関数 removeCache() を実行する．

図 3.7 にキャッシュ登録と転送処理のアルゴリズムを示す．

TCR で受信したパケットがコンテンツ検索要求 `SearchReq` またはダウンロード要求 `download-Req` の場合は，転送先ルータを選択するために図 3.9 に示す contentResolver() を実行する（図 3.7 の行 1）．このアルゴリズムは，3.4.2 節で述べる．

受信パケットがコンテンツ検索応答 `SearchRsp`，ダウンロード応答 `DownloadRsp` またはアドバタイズ要求 `AdvReq` の場合は，受信パケットのデータまたは送信先のキャッシュ可否を図 3.8 isCacheable() に示すアルゴリズムで調べ，更新可能な場合は，メソッド update() で該当するキャッシュを更新する（図 3.7 の行 9–10, 14–16, 17, 22–24, または，25）．

キャッシュが上限に達した場合は，update() 内でキャッシュ廃棄方式 DiscardMethod（3.2.1 節）により削除する．

受信パケットがアドバタイズ要求の場合は，受信したアドバタイズ要求に含まれるタグから転送先ルータを探し，転送先毎に視聴候補コンテンツリスト CL（ContentList の別名）を組み立て転送する（図 3.7 の行 21–31）．図 3.7 の行 26 では受信したコンテンツリスト CL の各タプルに含まれるタグ $row.Tag$ を持つタプルを RT（RouteByTag の別名）のメソッド match() により抽出し，該当するタプルから転送先ルータ $Router$ を得ている．

なお，type() は引数の型を得る関数で，send で始まる関数はパケットを生成して送信を行ない，関数 forward() はパケットを引数に向けて転送する．

検索要求またはダウンロード要求に対して自身が答えられる場合（図 3.7 の行 4 または行 6 が真）は，登録方式 EntryMethod（3.2.1 節）に応じて値を決め，関数 sendSearchRsp(), sendDownloadRsp() 内で，応答パケットのカウンタ cnt に設定する．方式が，LCE では経路長に対して十分大きな値を，LCD（Leave Copy Down）または MCD では，次段のルータのみキャッシュするために 1 を設定する．同様にアドバタイズ要求受信時にも関数 sendAdvReq() が cnt を設定する．

図 3.8 にキャッシュ可否判断のアルゴリズムを示す．

図 3.6 の行 1 で減じた cnt を，図 3.8 の行 1 で評価することで階層キャッシュ構成の登録方式を実現する．cnt が 0 以上の場合は，キー key に応じてポリシ（CCP または RCP）を選択し，ポリシにキーが存在すれば，そのコンテンツ情報または転送先ルータはキャッシュ可能である（図 3.8 の行 5）．

以上のように各 TCR がキャッシュ登録方式とポリシによりキャッシュ可否を判断する．

3.4.2 転送先ルータの経路選択

TCR の転送先ルータの経路選択アルゴリズム contentResolver() を図 3.9 に示す．

各 TCR は，コンテンツ検索要求 `SearchReq` またはダウンロード要求 `DownloadReq` パケットの転送先を図 3.9 により 1) 自身にデータを持つか，2) キャッシュ内に転送先のルータがあるか，3) 既定ルータ，4) SE または CMS の順で評価する．1) または 2) では，要求の引数がキャッシュデータのキーと一致するため次段の転送先ルータを一意に決定できる．

$defRouter$ は設定値の既定ルータで，$thisRouter$ は自身の IP アドレスである．関数 getListFromSE() は検索エンジンより視聴候補コンテンツのリストを，関数 getDataFromCMS() はコンテンツのオリジナルデータを持つ CMS よりコンテンツデータを取得する．

次に検索フェーズおよびダウンロードにおけるネットワーク全体の経路構成を説明する．図 3.7

で, 検索応答 `SearchRsp` を受信した TCR は, update() メソッドにより図 3.7 の行 10 で送信元を RT（RouteByTag の別名）へ，ダウンロード応答 `DownloadRsp` 図 3.7 の行 17 で RC（RouteByContent の別名）へ，それぞれキャッシュするため，同じ要求を受信した場合は転送せずに応答を返すことができる．以上から経路上ルータへのキャッシュ登録と転送先ルータの経路選択が実現され通信経路が短縮できる．この経路は，各 TCR のキャッシュの状態により各端末から各 CMS の経路上を伸縮する構成となる．

なお，Breadcrumbs [62] では，検索要求をマルチキャストして，応答が得られたルータをキャッシュし，Hop-aware BC [73] では最新の取得ノードへの誘導があるため，これらの検索経路は，CMS を頂点とするツリー構成となり，枝から下流方向のルータも対象となる．しかし，動画視聴に限定した場合は，視聴コンテンツの変動が大きいことから CMS 側に向けて検索する方法が有利となるので，TCR では，図 3.9 で示した転送先ルータの選択方法を採用し，転送先ルータが見つからない場合にオリジナルコンテンツを保持する上流側へ向けて転送を行なう．加えて，TCR ではキャッシュとコンテンツ検索の転送先ルータをタグにより決定することで，特定のコンテンツデータを取得するよりも，タグによる検索から，その時点でアクセス数の高いコンテンツを取得できるため，TCR の経路選択は動画視聴の配信基盤に向くと考える．

3.5 まとめ

TCR アーキテクチャの関連機器，内部構成，動作シーケンスを示し，タグをキーとして経路形成およびコンテンツ検索を行う方法について説明した．

経路形成および検索は各 TCR に設定するポリシにより挙動を変更することが可能である．ポリシの判断ロジックの追加により検討を省略したコンテンツアクセスの時間的変化の対応も可能と考える．

```
Input: recivedPacket as r, type(r.key) :=TagVURIVURI,BlkNo
Data: ContentList as CL, RouteByTag as RT
Data: ContentData as CD, RouteByContent as RC
Data: CachedData as CC, router as rtr
Result: Updating cache and sending packet(s)
if type(r)=SearchReqVDownloadReq then rtr ← contentResolver(key)
switch type(r) do
    case SearchReq:
        if rtr = thisRouter then sendSearchRsp(CL.match(key)) else forward(rtr)
    case DownloadReq:
        if rtr = thisRouter then sendDownloadRsp(requester) else forward(rtr)
    case SearchRsp:
        foreach i ∈ r.CL do
            if isCacheable(r, i.Tag) then CL.update(i.Tag, i.URI, thisRouter)
            if isCacheable(r, i.Tag) then RT.update(i.Tag, thisRouter)
        foward(requester)
    case DownloadRsp:
        key = (r.URI, r.BlkNo)
        if isCacheable(r, key) then
            CD.update(key, thisRouter)
            CC.update(key, r.data)
        if isCacheable(r, r.Tag, thisRouter) then RC.update(key, thisRouter)
        foward(requester)
    case AdvReq:
        target ← ∅
        foreach row ∈ r.CL do
            if isCacheable(r, row.Tag) then
                CL.update(row.Tag, row.URI, thisRouter)
                CD.update(row.URI, −1, r.from)
            if isCacheable(r, row.Tag, r.router) then RT.update(row.Tag, thisRouter)
            foreach t ∈ RT.match(raw.Tag)[Router] do
                if t ∈ target then t.CL ← t.CL ∪ row
                else
                    target ← target ∪ t
                    t.CL ← row
        foreach s ∈ target do sendAdvReq(s, s.CL)
```

図 3.7: キャッシュ登録と転送処理 cacheFowrad($recievedPacket, key$)

```
Input: recievedPacket as r
Input: type(key) := Tag ∨ Tag,Router
Output: cacheable or not
Data: ContentCachePolicy as CCP
Data: RoutingCachePolicy as RCP
if type(r) = (SearchRsp ∨ DownloadRsp ∨ AdvReq) ∧ r.cnt < 0 then return False
switch type(key) do
    case Tag: obj ← CCP
    case Tag,Router: obj ← RCP
if key ∈ obj.match(allow) then return True
else return False
```

図 3.8: キャッシュ可否判断 isCacheable($recievedPacket, key$)

```
Input: key:=[Tag|URI|<URI,BlkNo>]
Output: targetRouter
Data: ContentList as CL, RouteByTag as RT
Data: ContentData as CD, RouteByContent as RC
Data: CachedData as CC
obj ← Null
switch type(key) do
    case Tag :
        if key ∈ CL then obj ← CL
        else if key ∈ RT then obj ← RT
    case URI :
        if key ∈ CC then return this
        else if key ∈ CD then obj ← CD
        else if key ∈ RC then obj ← RC
if obj then return obj.match(key)[Router]
try forward(defRouter)
catch(timeout) then
    switch type(key) do
        case Tag : getListFromSE(key)
        case URI : getDataFromCMS(key)
```

図 3.9: 経路探索 contentResolver(key)

第4章　シミュレータの動作環境

近年，SNS（Social Networking Service）や CCN（Content-Centric Networking）のシミュレーションを行う機会が増えているが，既存のネットワークシミュレータには，アプリケーション層のサンプルコードが入手しにくいため研究者が記述する場合が多い．シミュレータの開発工数を削減するためには，1）言語の習得が容易，2）ライブラリが豊富，かつ，3）デバッグが容易，であることが求められる．Python 言語 [88] は，これらの要件を満たしている．1) は教育目的で利用されており，3) はインタプリタにより単純化したコードで動作確認できる．2) は，ソーシャルデータ [89]，集合知 [90] の例や 4.8.1 節で用いるツールがある．

シミュレーションのシナリオは，ネットワーク端に接続された視聴端末からネットワーク中央付近にあるサーバから動画コンテンツをダウンロードし，一定時間内に発生するダウンロード要求数が増加した場合の平均ダウンロード時間を測定する．

本章では，4.1 節で既存シミュレータの問題を，4.2 節でシミュレータの処理手順を述べ，4.3 節から 4.7 節で構成別のシミュレーション方法とシミュレーションパラメータを記す．4.8 節でシミュレータの実装について述べ，4.9 節でまとめる．

4.1　既存シミュレータの問題

ネットワークシミュレータの解説は文献 [91] に詳しい．従来のネットワークシミュレーションでは，伝送路やトランスポート層を対象としており，NS2 [92]，NS3 [93] や OmNeT++ [94] が用いられることが多い．

しかし，SNS（Social Networking Service）や CCN（Content-Centric Networking）を対象とする場合は，名前解決，キャッシュ探索，ランキング判定等のアプリケーション層の機能実装と同時に多数のルータ・ユーザ端末を対象とするために別の方法が求められる．

CCN や DTN の先行研究では，目的に応じてシミュレータを作成している．DTN の研究では OneSimulator [95] が，CCN の研究では，OMNeT++を拡張した ccsim [96] が用いられている．

タグをキーとしてコンテンツ検索およびコンテンツ情報およびデータのキャッシュを行うために，既存のシミュレータは利用できないため，新規に Python 言語でシミュレータを作成した．この

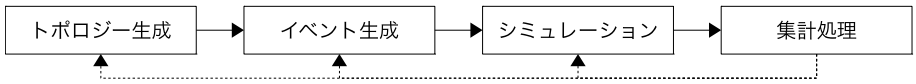


図 4.1: シミュレーションフロー

シミュレータは Python 言語により目的に応じて実装でき，コンテンツサーバやユーザ端末の動作も記述可能である．通信開始タイミングは，シミュレーション開始時に決定する必要がある．ルータ，サーバ，ユーザ端末のノード数は，実行するコンピュータの資源に依存するが，500 台程度を扱える．本実装での消費メモリ量はフラグメント単位のインスタンス生成のため容量を要している．シミュレーション実行時のアニメーション表示や高速化は今後の検討課題である．

コンテンツのキャッシュは，分散環境でキーによるオブジェクト管理である Key Value Store が利用でき Python 言語からは memcached [97] の利用が考えられる．ここでは，シミュレータは分散環境は利用せず，Python 言語のクラスにより機器を実装する．

4.2 シミュレータの処理手順

シミュレーションは，トポロジー生成，トランザクション生成，待ち行列シミュレーション，集計処理の順に行う．各フェーズごとにプログラムを作成し，オブジェクト整列化処理 `pickle` で永続化すると再試行が容易となる．

図 4.1 にシミュレーションの流れを示す．

シミュレータの処理手順を以下に示す．

1) トポロジー生成：機器を配置しトポロジーとリンク属性を設定する

2) イベント生成：イベント（時刻，視聴端末，コンテンツ）を生成する

3) シミュレーション：パラメータを変えて平均ダウンロード時間を測定する

4) 集計処理：結果から平均，標準偏差などを求め図化する

4.2.1 トポロジー生成

評価するネットワークを作成してサーバおよび視聴端末を配置し，各リンクに回線速度，MTU（Maximum Transmission Unit）を設定する．

インターネットのネットワークシミュレーションを行う場合に，GT-ITM [98] が用いられることが多い．しかし，TCR はアプリケーション層で動作するルータであり，要求数の増加に対する

平均ダウンロード速度を確認するため BA（Barabási-Albert）を使用する．

igraph [99] では，容易に各種トポロジーを生成できる．ルータ数 nR から成る BA グラフ g を作成し，ノード ID 10 から各ノード[1)]への最短経路をリスト list10 に格納し，図化する例を示す．

```
g = Graph.Barabasi(nR,directed=False)
list10 = g.get_all_shortest_pathes(10)
plot(g,file,layout=g.layout('fr'))
```

接続リンク数 (次数) が n であるノード集合は，g.vs.select(degree=n) で取得できるので，次数の大きなノードにサーバを，次数が 1 のノードに視聴端末を接続する．igraph では，ノード作成 (g.add_vertices) して接続 (g.add_eges) することで配置できる．g.degree(nodeID) で接続リンク数を取得できるため，回線速度および MTU は，リンク両端の次数から設定することが容易である．

ノード情報 g.vs，リンク情報 g.es は利用者が属性を追加できる．リンク ID が 1 の MTU と回線速度を設定する例を示す．#以降はコメントを表す．

```
g.es[1]['MTU'] = 1500   # unit in byte
g.es[1]['speed'] = 0.1  # unit in Gbps
```

図 4.2 にシミュレーションで用いたツリー状ネットワークの例を示す．この例では，TCR に対応したルータ数 100 で BA トポロジーを生成し，ノードの接続次数が多い順に 20 台のサーバを 1 台づつ接続している．サーバノードはオレンジ色で示している．リンク数が 1 のノード（ノードの色はグレイ）はエッジルータであるため，これに 300 台の視聴端末（ノードの色は白）を均等に接続している．各ノードは接続次数により，2：シアン，3：マゼンタ，4：黄色，5：青，6：緑，7 以上を赤で表している．

回線属性は MTU と回線速度のタプルで，リンク両端の接続次数の小さい方の値により 5 レベルに振り分ける．図中のコードはその振り分け処理である．

4.2.2 イベント生成

イベントでの時刻，要求端末，対象コンテンツなどの選定は，乱数や統計にとる分布を使用するが，Python 言語では科学技術ライブラリ SciPy [100] が有用である．

待ち行列ライブラリ SimPy [101] では，送受信要求を開始する時間を事前に設定する方法が主流である．端末，ファイル，および開始時刻を統計分布より生成する．

[1)] igraph では vertex，edge を用いているが，ノード，リンクと同義として扱う．

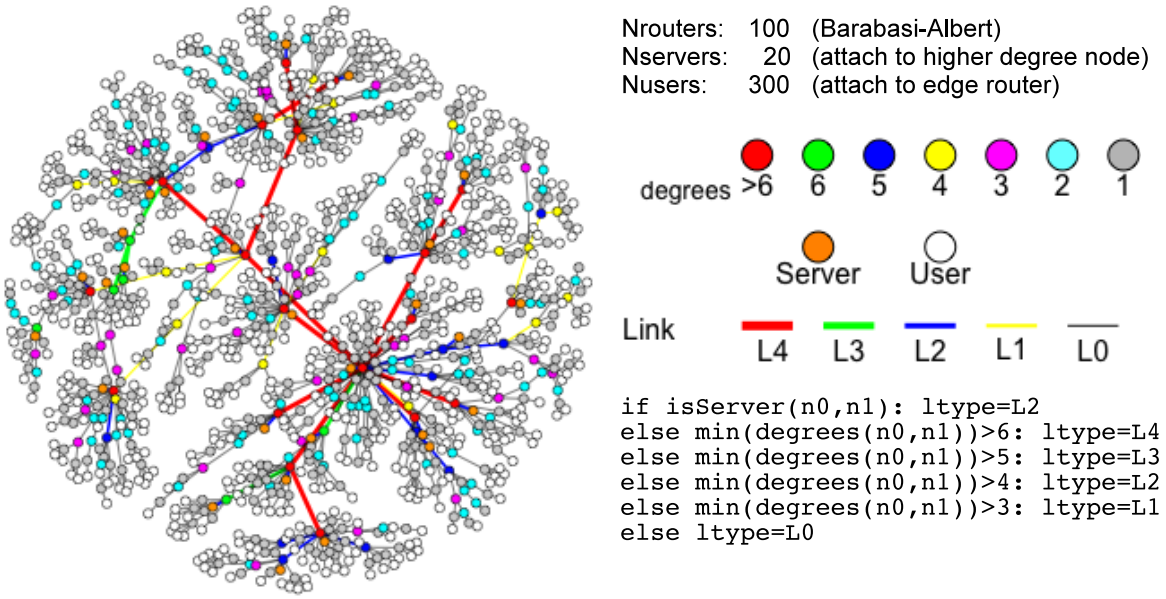


図 4.2: ツリー状ネットワーク（BA: Barabási-Albert）

表 4.1: タグランキング分布（d, N=100, K=20）

discount d	top1	top2	top4	top20	over80%
0%	0.58	0.74	0.86	0.98	rank=2
2%	0.27	0.35	0.42	0.56	rank=46
4%	0.18	0.23	0.28	0.43	rank=57
6%	0.13	0.17	0.21	0.36	rank=62

表 4.1 にイベント生成で作成したタグ分布の例を示す．

タグを N=100 種類用意し，ランキング 1 位から K=20 位までの分布を作成する．ランキング第 k 位の出現頻度は式 2.7 より求める．シミュレーションでは d=$0, 2, 4, or 6\%$ を使用する．d=6%では，各タグの生起確率は，ほぼ同程度となる．$\mathrm{D}_{k;d,K}$ の総和を分母として正規化し，ダウンロード要求数 $nreqs$ との積でタグアクセス数を算出する．

表 4.1 では，割引率 d=2%のとき全体の 80%がランキング 2 位まででヒットするが，d=2%になると全体の 80%を占めるには，ランキング 46 位までを要している．

図 4.3 は，表 4.1 を実数で計算した連続値として図示している．左図では，ランキング 100 位までを，右図ではランキング 20 位までの部分を表示している．左図で第 21 位から 100 位までの生起確率は式 2.7 により同一であるため図で傾きは一定に近くとなっている．割引率 d が 5%にする

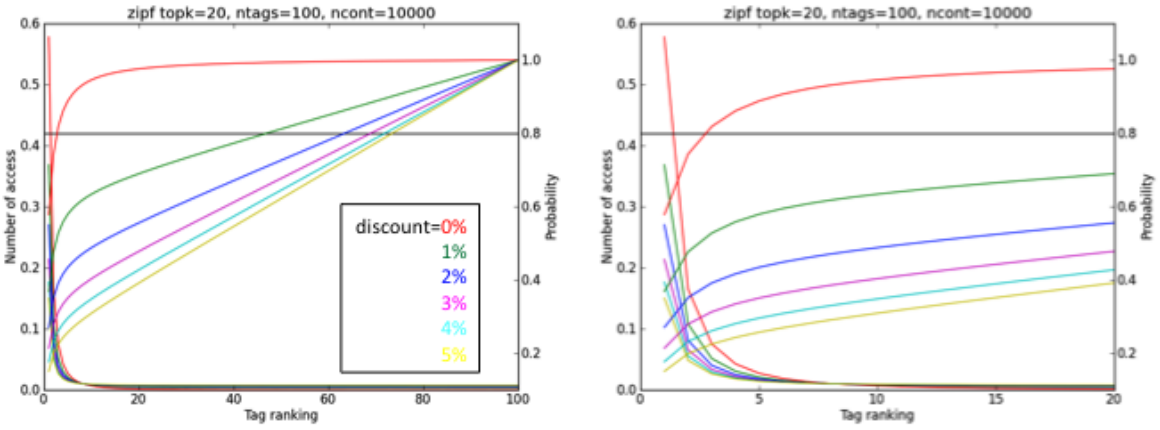


図 4.3: タグアクセス分布

と，生起確率はほぼ一様となり，第 20 位までの生起確率は，40%程度である．実際のシミュレーションでは，式 2.7 をランキング順位を整数値として使用する．

評価用に生成するイベントで，視聴者が判断する要素は現在のシミュレータでは未対応であるため，以下のイベントは生成せず評価にも用いない．

1. アドバタイズメント要求
2. 視聴端末でのコンテンツキャッシュと Carry and Forward
3. Closed area network と視聴者による Scatter 機能

4.2.3 シミュレーション

転送トランザクション

4.2.1 節で作成したトポロジー，4.2.2 節で生成したイベントから，開始時刻に応じて待ち行列ライブラリ SimPy の機能により転送トランザクション（インスタンス）を生成する．

キューイングはパケットを管理単位として実装するため，転送トランザクションは MTU 単位のサブトランザクションを生成する．サブトランザクション（サブインスタンス）は，送信元ノードから通信経路単位でルータに順次キューイングし，目的ノードに到達するまでの時間を `now()` 関数により取得する．このとき各 TCR は，TCR アーキテクチャの経路探索とキューイングを実施を行う．

最後のデータを含むサブトランザクションが要求元に達した場合にがトランザクションは完了する．

キューイングは，データ転送順序を確保するためサブトランザクションでシーケンス制御を行う．各リンク毎に MTU と回線速度を考慮し，送信データが MTU を超える場合は，パケットを分割して送信する．

待ち行列の処理

待ち行列の実装例を示す．SimPy は，待ち行列を処理するライブラリで，キューを作成し，キューイング，待ち処理，開放を行うことで待ち行列処理を行うことができる．以下に概略を示す．

各ノードの接続されたリンクに関数 `Resource` でキューを作成する．

待ち行列処理は，1）`request(q)` でキューイングし，2）`hold(w)` で `w` 時間キューを占有し，3）`release(q)` で資源を開放する．ここで `wait` は転送に要する時間（MTU/回線速度）である．これらの前後に関数 `SimPy.now()` により時刻を取得することで転送に要した時間が得られ，集計用にファイル出力する．SimPy ではキュー長の統計情報も取得できる．

以下に概略を示す．

```
Resource(capacity=1, name=q)   # キュー作成
t1 = now()                     # 開始時刻
yield request, self, q         # キューイング
yield hold, self, wait         # キュー占有
yield release, self, q         # キュー開放
t2 = now()                     # 終了時刻
```

4.2.4 集計処理

シミュレータの出力を読み込み R 言語 [102] の入力ファイルを生成する．生成されたファイルを実行し，シミュレーション結果の統計情報や図を得る．

集計対象について説明する．各ルータのキャッシュが一通りアクセスされた場合が定常状態と想定する場合がある．本評価では，人気コンテンツの変遷によりキャッシュが全くヒットしない場合も定常的に発生すると考え集計に含めている．データダウンロード数が 2 万程度に対してキャッシュエントリ数，コンテンツ数，およびタグ数は 100 程度であり，初期結果を含めても結果への影響は小さいと考えてキャッシュが未使用である状態から集計を行う．

表 4.2: シミュレータのパラメータと処理の関係

項目	変数名	T[a]	E[b]	S[c]	値の例
TCR ルータ数	$nRouters$	*	*	*	100
CMS サーバ数	$nServers$	*	*	*	20
視聴端末数	$nViewers$	*	*	*	300
ダウンロード要求数	$nreqs$	-	*	*	2000, $\cdots$ 20,000
タグ数	N	-	*	*	100
タグ割当数	$ntag$	-	*	*	4
トップ K	K	-	*	*	20
割引率	d	-	*	*	2%
動作モード	$mode$	-	-	*	TCP|URI|Tag
キャッシュ登録方法	$entryMethod$	-	-	*	LCE|LCD|MCD
キャッシュ廃棄方法	$discardMethod$	-	-	*	LRU|FIFO
キャッシュ容量比	$cache\%$	-	-	*	20%

[a]Topology generator
[b]Event generator
[c]Queuing Simulator

4.2.5 主要なシミュレーションパラメータ

表 4.2 に主要なシミュレーションパラメータと使用する処理の関係を示す．T, E, S 列の * は，それぞれトポロジー生成，イベント生成，シミュレーションでパラメータを使用することを，-は使用しないことを表す．

トポロジー生成では，ノード数の情報を用いる．続くイベント生成では，加えてダウンロード要求数によりダウンロードを行う視聴端末，時刻を決定し，対象動画コンテンツの選択のために，タグ数．タグ割当数，割引率などを用いる．シミュレーションでは，加えて TCR の動作，キャッシュ登録および廃棄方法を指定する．

4.3 タンデムモデル

4.3.1 評価の目的

コンテンツのデータサイズが比較的大きな動画ファイルのダウンロードにおいてキャッシュが機能するか確認する．ルータでは，コンテンツを固定長のチャンクによりキャッシュを行う．

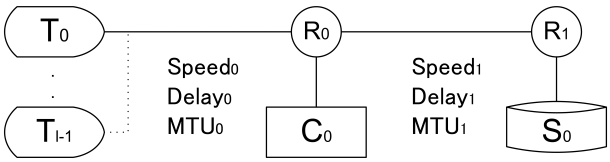


図 4.4: タンデムモデル

4.3.2 評価方法

平均ダウンロード時間をキャッシュする場合としない場合で比較する．図 4.4 に評価用ネットワークを示す，

端末 T_0 から T_{l-1} が CMS ノード S_0 からコンテンツを取得する．単位時間に一定のアクセス数があるとキャッシュサーバ C_0 がコンテンツの固定長ブロックを蓄積する．ルータ R_0，R_1 で，MTU 単位で待ち行列により待ち時間が発生する．待ち時間は，MTU サイズと回線速度 $Speed_n$ により決定する．MTU がウインドウサイズに達した場合に遅延 $Delay_n$ が発生する．

端末から生成されるトランザクションは，CMS 上の任意のコンテンツを選択し，シミュレーション時間内にランダムに生成する．特定のコンテンツが集中的に発生する状況を評価トランザクションとして生成する．

コンテンツ数は，50 種類でサイズはすべて 100MB とした．

端末 T とキャッシュサーバ C_0 間のリンクは，$Speed_0 = 100\text{Mbps}$, $\text{Delay}_0 = 5\text{ms}$, $\text{MTU}_0 = 1500\text{octet}$ とした．

キャッシュサーバ C_0 と CMS ノード S_0 間のリンクは, $Speed_1 = 1\text{Gbps}$, $\text{Delay}_1 = 100\text{ms}$, $\text{MTU}_1 = 9000\text{octet}$ とした．

Window サイズはともに 10 パケットとした．

遅延時間は，インターネットのラウンドトリップタイムを調査した文献では 50ms 程度の報告があり [103]，動画による負荷増加を想定し $Delay_1$ は 100ms を用いた．

キャッシュサーバ C_0 では，データを 1MB の固定長ブロックとして管理し，キャッシュ容量は 1000 ブロックとした．

トランザクションは，1 時間に 100 から 1000 まで増分 100 で発生し，いずれの場合にも特定のコンテンツをアクセスするトランザクションを開始 20 分から 30 分の間に少なくても 8 トランザクションが生成される．

端末側からキャッシュ上にデータが存在するか固定長ブロック毎にチェックする．コンテンツの固定長ブロックが単位時間内に一定以上アクセスされた場合に，キャッシュサーバは，CMS よりデータを取得する．

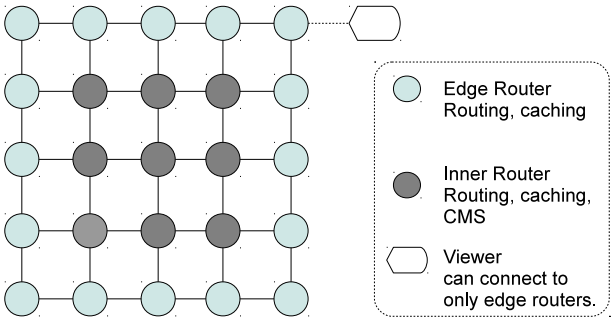


図 4.5: 格子状モデル

シミュレーションでの単位時間は 10 分とし，取得を判断するアクセス数のしきい値 numAccess は 3, 5, 7 とした．一度に取得する連続ブロック数 loadBlocks は 10, 5, 3 とした．

表 4.3 に評価パラメータをまとめる．

4.4 格子状ネットワークモデル

4.4.1 評価の目的

ネットワークの規模により経路上のルータでキャッシュ有無の比較を行う．

コンテンツを取得する際にタグを使用する割合（タグ検索率）を変えて性能向上の効果を確認する．

端的なキャッシュルータネットワークの例として，格子状に接続されたキャッシュルータのトポロジーを想定する．これは，データセンタなど CMS が集中した場所に複数ある状態を想定している．また，格子状ネットワークでは，視聴端末から CMS までの経路長の差が少ないため，キャッシュ有無の性能差を把握しやすいメリットがある．

4.4.2 評価方法

図 4.5 に評価用格子状ネットワークを示す．ネットワークトポロジーは，ルータを格子状に配置し，外側（リンクが 4 未満）のエッジルータに視聴端末 Viewer を接続する．内側（リンクが 4 本）の内部ルータは，CMS の機能と TCR の機能を持つことができる．

コンテンツ，端末の割当先は一様乱数によりノードを決定する．

初めに，キャッシュを行わない構成で，格子状ネットワークを格子サイズを 3×3，5×5，7×7，9×9 と変えて，負荷の増加とともに平均ダウンロード時間が急変する格子サイズを確認する．

次に，影響が確認できた格子サイズでルータでのキャッシュ有無の影響を確認する．

リンク速度，遅延は，視聴端末・エッジルータ間が 100Mbps，2ms で，ルータ間は，100Mbps，10ms とする．

コンテンツ数 1000，コンテンツのデータサイズ 100MB，タグ数 1000，各コンテンツに平均 4 つのタグを一様乱数により割り当てる，

表 4.3 に評価パラメータをまとめる．

4.5 ツリー状ネットワークモデル

4.5.1 評価の目的

性能評価はネットワーク全体を対象として，ダウンロード時間から応答性を，ダウンロード時間の標準偏差から安定性を評価する．性能評価での総キャッシュ量から経済性を総合評価で用いる．シミュレーションは，測定条件ごとに n=5 回実施して平均値を採用する．本評価では，全視聴端末での合計ダウンロードデータ量は時間あたりテラバイトオーダであり，アドバタイズ要求や検索応答のパケットによる性能への影響は，ダウンロード応答に比べて軽微であるため，それらのパケット発行は省略する．同様にコンテンツメタ情報は，コンテンツデータに比べて十分小さいため，キャッシュ容量の集計から除外する．

配信方式は，ルータでのキャッシュ有無とキャッシュキーによる相違から定義される．

Traditional 方式 キャッシュは使用しない

URI-based 方式 URI をキーとしたキャッシュ

Tag-based 方式（TCR アーキテクチャ） タグをキーとしたキャッシュ

以下に可変パラメータの定義を示す．割引率 d は 4.5.2 節のタグの分布で詳細を説明する．

ダウンロード要求数 $nreqs$： 総ダウンロード要求数

タグ数 $ntag$： 1 つのコンテンツに付与するタグの数

割引率 d： タグランキング分布のロングテールの程度

キャッシュ容量比 $cache\%$： 1 つのルータのキャッシュ容量/総コンテンツ容量

4.5.2 シミュレーションの条件

表 4.3 にシミュレーションパラメータを示し，シミュレーション条件についてネットワーク，コンテンツの割当，タグ分布，トラヒック，キャッシングの項目別に説明する．

ネットワーク

TCR ルータ数 $nRouters$=100 の BA（Barabási-Albert） モデル [86] のトポロジーを作成する．CMS サーバ数 $nServers$=20 の CMS サーバを，各ルータノードのリンク数（接続次数）が高い順に各 1 台接続する．Viewer 端末数 $nUsers$=300 の Viewer を接続次数が 1 のルータに均等に接続する．各リンクにおいて両端のノードの接続次数が双方ともに Lth=6 である場合を境として異なる回線速度，MTU（Maximum Transmission Unit）を選ぶ．CMS，TCR の各リンクにキューを設け，MTU 単位でキューイングし，送信されるまで，回線速度 ×MTU の時間だけ待つ．パケットの順序性は各 TCR でのフロー制御により保証する．パケットの紛失や廃棄は，起きない前提とする．

ルータ数は，国内にある CMS サーバを国内から視聴する場合を想定している．インターネットの端末から Web サーバ迄の平均経路長の中央値は，米国西部で 12，日本では 19 であった [104]．この報告は，米国西部を中心として測定しているので，日本国内に CMS があり国内から視聴する場合の平均経路長は 7 程度である．$nRouters$=100 の BA ネットワークの平均経路長は 7.2 程度であるため，評価するネットワークは，国内での視聴環境に近いモデルである．

図 4.6 に評価用トポロジー生成方法の例を示す．

行 1 では，$nRouters$ 個のノードから成る BA トポロジーを生成している．行 2 の関数 getTopN-Degree() では，BA グラフのノードの接続次数が高い順に $nServers$ 個のルータノードを得ている．行 4 では行 3 で得られた高接続ノードのルータに 1 つづつサーバを接続するリンクを生成している．行 5 ではメソッド filter によりルータかつ接続次数が 1 のノードを抽出している．行 7 では行 5 で得られた接続対象のルータをランダムに選択して，視聴端末を接続している．関数 getDegree() はリンク両端の接続次数を返す．行 9 では，接続次数の小さい方が 6 以上か判定している．行 10，13 ではリンクの MTU を，行 11，14 ではリンクの速度を設定している．

コンテンツの割当

コンテンツを $nContents$=100 種類用意し，CMS に一様乱数により割り当てる．コンテンツ配信方法が Tag-based の場合に各コンテンツにタグを重複しないように $ntag$=$1, 2, 4, or 8$ 個割り当てる．$ntag$ の値は，平均値である．

```
Input: nRouters, nServers, nViewers
Result: Graph := (Link, Node)
// make link higher degree router from/to server
Graph ← BA(nRouters)
list1 ← getTopNDegree(Node, nServers)
for i ∈ nServers do
    makeLink(list1_i, i)
// make link edge router from/to viewer
list2 ← Node.filter(type = router ∧ degrees = 1)
for j ∈ nViewers do
    makeLink(random(list2), j)
// assign link attributes (MTU, speed)
for k ∈ Link do
    if min(getDegree(k)) ≥ 6 then
        k.mtu ← 3Kbyte
        k.speed ← 1Gbps
    else
        k.mtu ← 1.5Kbyte
        k.speed ← 100Mbps
```

図 4.6: トポロジー生成方法 topologyGen($nRouters, nServers, nViewers$)

タグの分布

スケールフリー性からアクセスの少ないタグがロングテール状に存在するが，割引率 d は，ロングテールの程度を表す．タグを N=100 種類用意し，ランキング 1 位から K=20 位までの分布を作成する．ランキング第 k 位の出現頻度は式 2.7 より求める．

式 2.7 の $\sum_{n=1}^{k} \frac{1}{n}$ 部分は，Zipf 関数である．割引率 d はランキング第 1 位を 1 とした比であり，K 位から N 位までのランクは d の生起頻度となる．d が大きいほど上位ランクのアクセス頻度が少なくなる．ロングテール部分の長さをタグ数 N で，生起確率を d で実際の分布と比較しやすくする．シミュレーションでは d=$0, 2, 4, or 6\%$ を使用する．d=6%では，各タグの生起確率は，ほぼ同程度となる．$\mathrm{D}_{k;d,K}$ の総和を分母として正規化し，ダウンロード要求数 $nreqs$ との積でタグアクセス数を算出する．

トラヒック

総ダウンロード要求数 $nreqs$（2,000 から 16,000 まで増分 2,000）のコンテンツ取得をシミュレーション時間 $simTime$=3,600 秒以内の任意の時刻より任意の Viewer 端末から生成する．コンテンツデータ取得開始から完了までの時間と経路長を測定する．ダウンロード要求あたりコンテンツデータサイズ $dataSize$=250MByte のダウンロードを行う．全 Viewer がダウンロードするデータ量は，$nreqs$=16,000 で 4TByte となり，総トラヒック量は，$nreqs \times dataSize \times$ 平均経路長である．ダウンロード要求を行う Viewer，対象コンテンツ，開始時刻は，一様乱数により生成する．配信方式が Tag-based の場合にタグのアクセス頻度分布は，式 2.7 に従う．

キャッシング

Traditional の場合は，キャッシュは使用しない．URI-based，Tag-based の場合は，キャッシュ処理を行う．キャッシュ容量比 $cache\%$ は URI-based の場合は 100%とし，Tag-based の場合は，$4, 8, or 12\%$ を用いる．各 TCR でダウンロード応答を中継する際に，キャッシュポリシによりタグ，URI, コンテンツをキャッシュする．キャッシュが容量を超えた場合の廃棄方法は LRU を用いる．

4.6 階層キャッシュ方式による影響

4.6.1 評価の目的

先行研究で評価されているキャッシュ登録方式とキャッシュ廃棄方式がタグをキーとした場合にどの程度影響するか確認する．

4.6.2 評価方法

4.5 節のツリー状ネットワークの場合の，キャッシュ登録方式（LCE, LCD, MCD），キャッシュ廃棄方式（LRU, FIFO）による平均ダウンロード時間の相違を確認する．

文献 [105] の評価では，階層キャッシュの登録方式が LCD, LCE(Always) より random（固定確率，正規分布）がキャッシュヒットは 2〜3%高い報告がある．ここで経路が 1 つの場合のキャッシュ廃棄方法は，LRU のキャッシュヒット率が一番高いが，経路が複数ある場合は正規分布を用いた方が効率は良い．評価では，階層キャッシュ登録方式 LCD, LCE, MCD や LRU, FIFO の影響

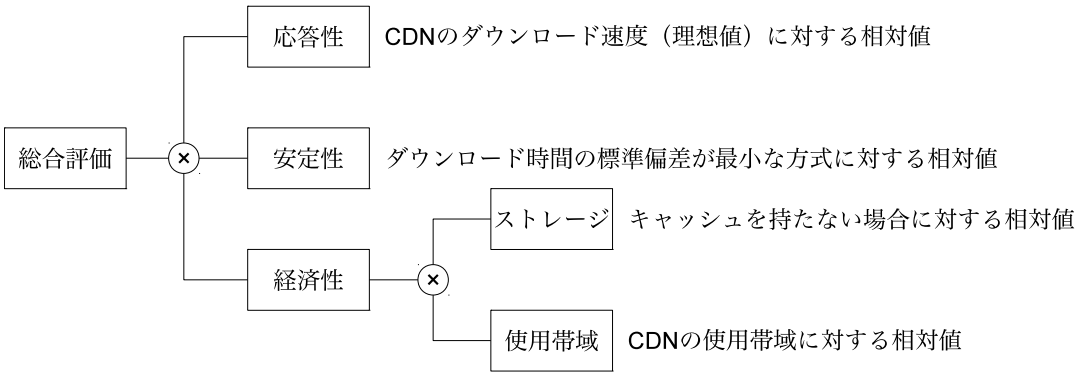


図 4.7: 総合評価の構成

を見たが，タグアクセスランキングの生起確率はランキング順位が下がると急激に低下することから，random 方式での階層キャッシュ登録・廃棄は不利と考えて省略する．

4.7 総合評価

4.7.1 評価の目的

総合評価の目的は，情報流通方式を総合的に評価し，TCR アーキテクチャが他方式に対する優位性を定量的に評価することである．

4.7.2 評価方法

5.1.3 節で用いたツリー状ネットワーク構成での性能評価結果およびシミュレーションで用いた表 4.3 のシミュレーションパラメータ値から算出する．比較する方式は，以下の 3 種である．

Traditional： 通信経路上でキャッシュは利用できない

URI-based： 通信経路上で URI によりキャッシュおよびルーティングを行う

Tag-based： 通信経路上でタグによりキャッシュおよびルーティングを行う

URI-based の場合にキャッシュ容量比 $cache\%$ を，Tag-based の場合，タグ割当数 $ntag$, 割引率 d, キャッシュ容量比 $cache\%$ をパラメータとする．

総合評価の構成を図 4.7 に示す．総合評価は，応答性，安定性，および経済性の 3 つの指標から構成され，各項の総積である．経済性は，ストレージ容量と使用帯域の要素があり，各要素の積

により求める．各項は，最善の場合を 1 とした比であるため，理想状態では総合評価値は 1 となる．総合評価を式 4.1 に示す．

$$\begin{aligned} \text{総合評価} &= \text{応答性} \times \text{安定性} \times \text{経済性} \\ &= \text{応答性} \times \text{安定性} \times \text{ストレージ経済性} \times \text{使用帯域経済性} \end{aligned} \tag{4.1}$$

応答性の理想値は，端末に隣接したルータからコンテンツをダウンロードし，他の通信トラヒックが無い場合である．これは，CDN で端末から 1 ホップのルータにコンテンツが蓄積され，ダウンロードの通信で遅延が全く無い状態と等価となる．5.1.3 節の環境では，表 4.3 よりコンテンツのダウンロード時間は，250MB/100Mbps$= 250 \times 10^6 \times 8/(100 \times 10^6) = 20$[sec] となる．この値を基準値として，式 4.2 により相対値を求める．

$$\text{応答性} = \frac{CDN\ \text{でのダウンロード時間の理想値}}{\text{各方式のダウンロード時間}} \tag{4.2}$$

安定性では，各方式でのダウンロード時間の標準偏差が最小の値を理想値として式 4.3 により相対値を求める．応答性を求めた式 4.2 では，CDN の理想値を基準としたが，通信回線上の競合が無い場合を想定しているため，安定性での基準値は各方式のうち最善のものを基準とする．

$$\text{安定性} = \frac{\min(\text{各方式のダウンロード時間の標準偏差})}{\text{各方式のダウンロード時間の標準偏差}} \tag{4.3}$$

経済性はネットワーク全体で使用する資源を評価する．経済性は式 4.4 に示すようにストレージ経済性と使用帯域経済性の積である．

$$\text{経済性} = \text{ストレージ経済性} \times \text{使用帯域経済性} \tag{4.4}$$

ストレージ経済性は，各方式で使用するネットワーク全体のストレージコストであり，ストレージ容量には，CMS のストレージと各ルータのキャッシュが含まれる．最小の方式の容量を基準とし，使用ストレージ容量が増えるほど経済性が悪化する．Traditional 方式は，通信経路上のルータにキャッシュを持たないため，使用ストレージ容量は他方式に比べ少ない．各ルータのキャッシュは未使用部分があれば，その容量を除外して使用した部分のストレージ容量から経済性を算出するべきである．今回の評価では，予備試験によりすべてのルータのキャッシュは使用されていることを確認しているため，各方式のストレージ経済性はダウンロード要求数によらず一定である．表 4.3 より，では CMS に保管されるオリジナルコンテンツ数は 100 で，コンテンツサイズは，250MB であるため，使用ストレージは，100×250MB となる．ストレージ経済性は，この値

をストレージ容量の基準として式 4.5 により求める．

$$\text{ストレージ経済性} = \frac{\min(\text{各方式の使用ストレージ容量})}{\text{各方式の使用ストレージ容量}} \tag{4.5}$$

使用帯域経済性は，ダウンロード通信で使用した回線の通信帯域総量の相対値で，基準は，視聴端末に接したルータからダウンロードに使用する帯域である．これは，CDN での使用帯域に相当する．各方式のダウンロード性能をシミュレーションする際に，端末からコンテンツを取得したルータまたは CMS までの経路長と回線速度がわかるため，各リンクの回線速度とパケット数の積の総和によりネットワーク上の使用帯域が式 4.6 より求められる．

$$\text{使用帯域経済性} = \frac{CDN\text{ での回線速度} \times \text{転送データ量}}{\text{各方式の} \sum_{link=1}^{\text{経路長}} \text{回線速度} \times \text{転送データ量}} \tag{4.6}$$

表 4.3 より端末と接した回線の回線速度は 100Mbps, その他は 1Gbps なので，シミュレーションでのネットワーク構成での使用帯域経済性は，式 4.7 となる．ここで平均経路長は，ダウンロード要求数 $nreqs$，タグ数 $ntag$，割引率 d，またはキャッシュ容量比 $cache\%$ などのシミュレーションパラメータにより異なる結果となる．

$$\begin{aligned} \text{評価ネットワークでの使用帯域経済性} &= \frac{100Mbps}{100Mbps + 1Gbps \times (\text{各方式の平均経路長} - 1)} \\ &= \frac{1}{1 + 10 \times (\text{各方式の平均経路長} - 1)} \end{aligned} \tag{4.7}$$

総合評価の求め方を述べた．

4.8 実装

4.8.1 使用ツール

シミュレーションで使用した言語，ライブラリは，言語：Python [88]．待ち行列：SimPy [101]，トポロジー生成：igraph [99]，科学技術計算：SciPy [100]，O/R マッパ：SQLAlchemy [106]，SQL データベース：SQLite3 [107] 集計・図化：R [102] などを用いた．

4.8.2 ハードウェア

CPU: Intel Core i5，メモリ: 32GB の 64 ビット Linux 機でシミュレータ開発およびシミュレーションを実行した．

4.8.3 プログラム規模

開発したシミュレータソフトウェアの規模は，Python 言語で，トポロジー生成に約 300 行，エベント生成約 300 行，シミュレーション約 600 行程度で，実行や集計用のスクリプト類が計 2000 行程度である．

4.9 まとめ

評価で使用するシミュレータの処理の流れと機能を説明した．段階的に実施する各シミュレーションの目的，評価条件，パラメータを解説し，シミュレータ環境および開発規模を記した．

表 4.3: シミュレーションパラメータと値

Category	parameter	tandem	matrix	tree
Network	Topology	tandem	matrix25	BA100
	$nRouters$	1	25	100
	$nServers$	1	9	20
	$nUsers$	1	16	300
	Link speed 1	1Gbps[a]	100Mbps[c]	1Gbps[e]
	Link speed 2	100Mbps[b]	100Mbps[d]	100Mbps[f]
	Link MTU 1	9KB[a]	10KB[g]	3KB[e]
	Link MTU 2	1.5KB[b]	10KB[g]	1.5KB[f]
Queue	length	∞	∞	∞
	multiplicity	1	1	1
	delay 1	5ms[a]	N/A	N/A
	delay 2	100ms[b]	N/A	N/A
Content	$nContents$	50	1000	100
	$dataSize$	100MB	100MB	250MByte
	assign to CMS	random	random	random
Tag	N	N/A	1000	100
	K	N/A	N/A	20
	$ntag$	N/A	4	$1, 2, 4, or\ 8$
	Tag distribution	N/A	constant	D function (Zipf-like)
	d (discount rate)	N/A	N/A	$0, 2, 4, or\ 6\%$
Caching	Entry method	LCE	LCE	LCE
	Discard Method	LRU	LRU	LRU
	$cache\%$	N/A	N/A	Tag:4,8,or12%,URI:100%
	Routing method	shortest	shortest	shortest (initial: DNS)
	ContentCachePolicy	N/A	N/A	'allow,all'
	RoutingCachePolicy	N/A	N/A	'allow,all,all'
	chank size	1MB	1MB	1MB
Event	$nreq$	$\leq$1k	$\leq$22k	$\leq$16k
	$simTime$	3600sec	3600sec	3600sec
	Requester	fix	random	random
	Search key	URI	Tag or URI	Tag or URI
	tagsearch%	0	0 to 80 by 20	0 or 100

[a]router from/to CMS
[b]viewer from/to router
[c]router from/to router
[d]viewer from/to router
[e]$L_{th} \geq 6$
[f]$L_{th} < 6$
[g]chunk サイズまで連続送信

第5章　結果と分析

本章では，第 4 章の評価方法に対応して結果を提示し分析を行う．初めにネットワーク構成の評価結果を，5.1 節でタンデムモデル，格子状モデル，ツリー状モデルの順に提示する．次に先行研究にある階層キャッシュの影響を 5.2 節で確認する．最後に 5.3 節で総合評価を提示して，まとめる．

5.1　ネットワーク構成と規模による評価

5.1.1　タンデムモデルでの性能評価

図 5.1 に評価用の対象コンテンツとその他のコンテンツのキャッシング有無による平均ダウンロード時間を示す．

シミュレーション時間で 1 時間あたりのダウンロード数を増やした場合にキャッシュしない場合（赤線）は，ダウンロード時間は 9.8 秒程度であるが，キャッシュを行う場合（青線）ではダウンロード数を増やすとダウンロード時間は 8.2 秒に低下し，キャッシュの効果が得られた．

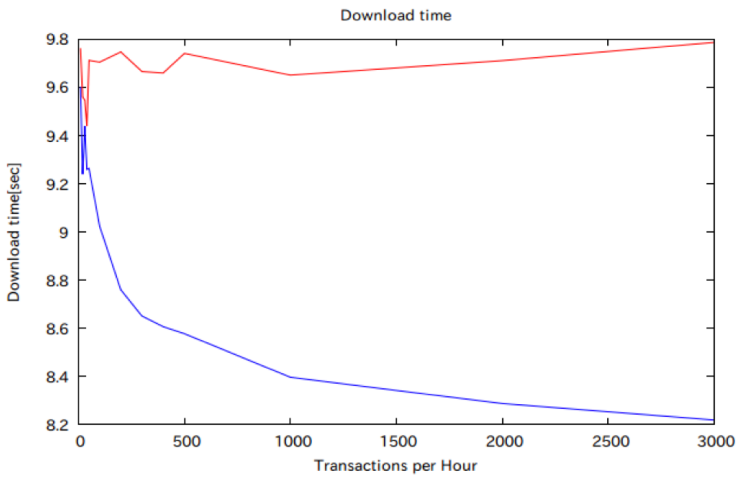


図 5.1: タンデム構成：キャッシュ有無のダウンロード時間

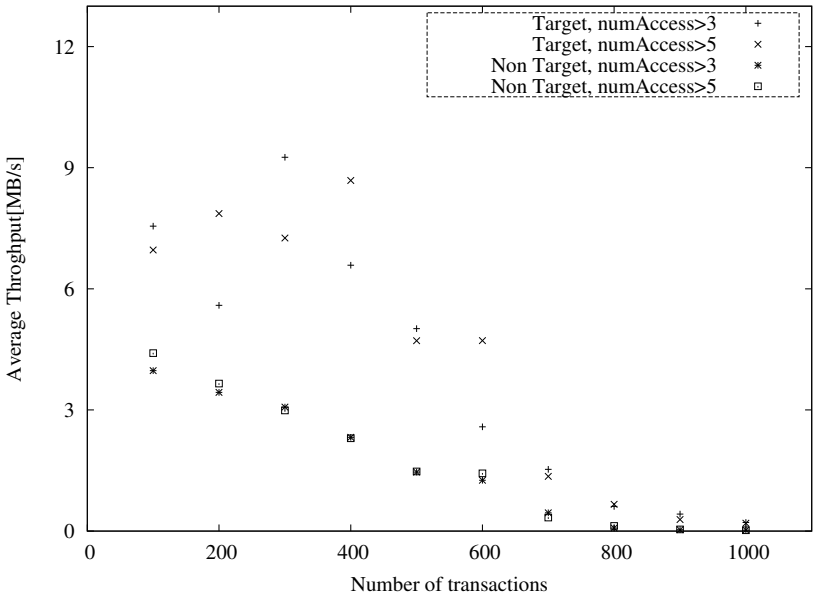


図 5.2: タンデム構成：同一コンテンツの集中アクセスによる平均ダウンロード速度

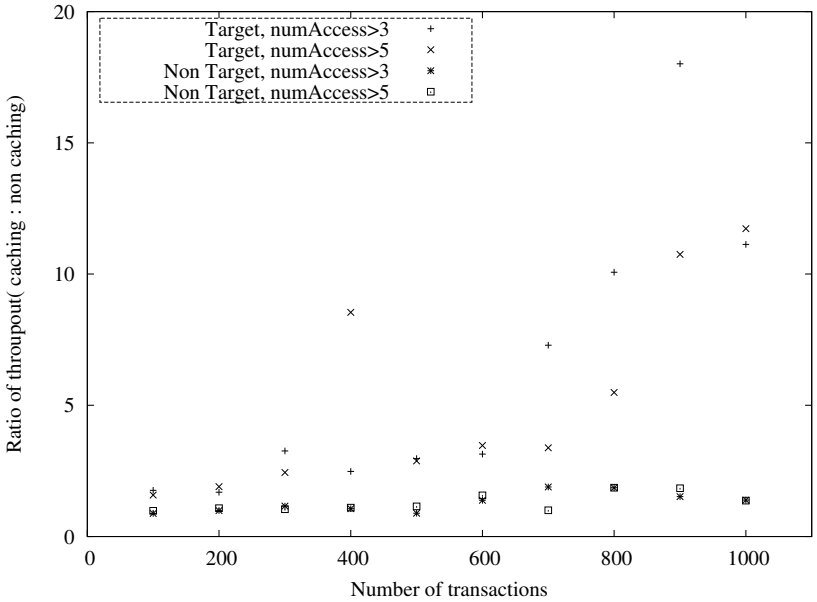


図 5.3: タンデム構成：キャッシュ有無による性能比

図 5.2 にキャッシング有効時のスループットを示す．図 5.1 では，平均ダウンロードで表示しているが，この図では，ダウンロードするコンテンツが集中する場合（Target）とランダムな場合（Non Target）の相違を見る．numAccess はアクセスされた回数によりしきい値を 4 以上と 6 以上で集計対象を分けている．

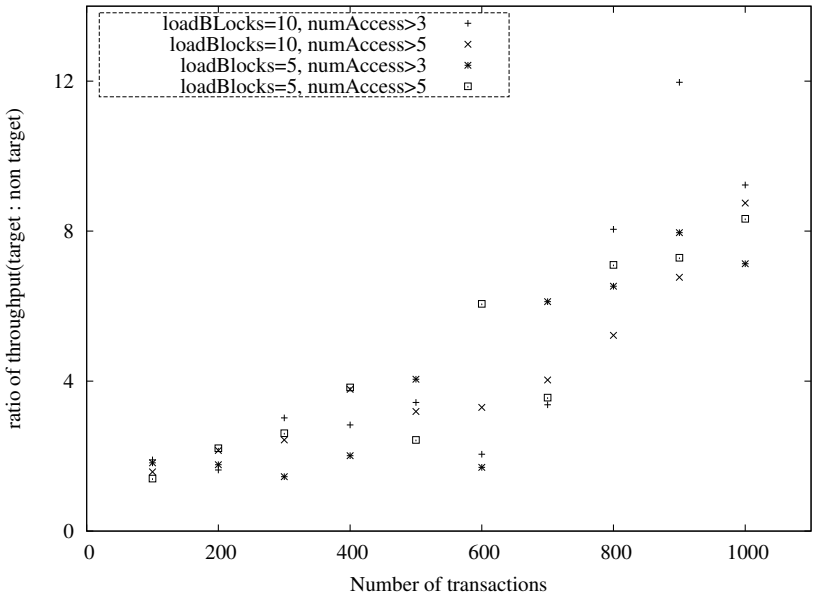


図 5.4: タンデム構成：対象コンテンツと一般コンテンツの性能比

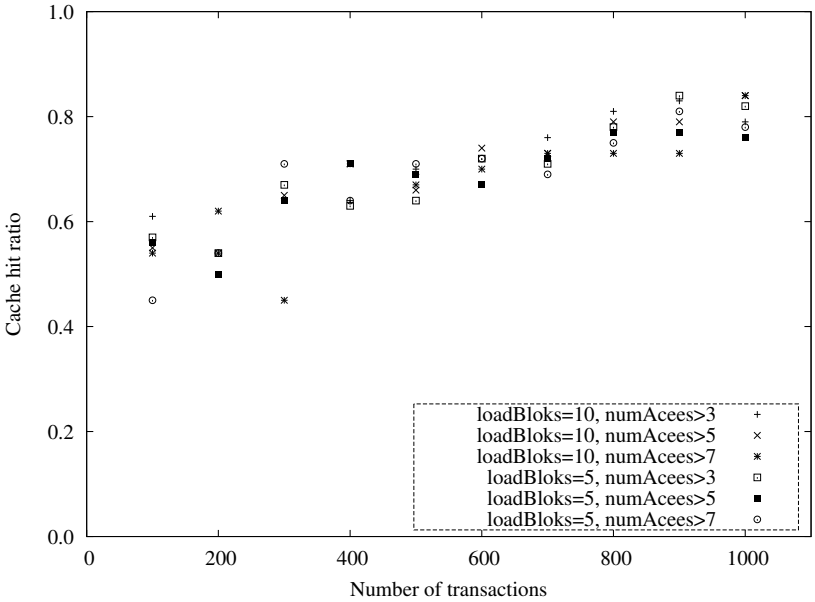


図 5.5: タンデム構成：対象コンテンツのキャッシュヒット率

ダウンロード要求アクセスがランダム（Non Tatget）ではトランザクション数に応じて性能が低下するが，アクセスが集中する対象トランザクションではキャッシングが機能してトランザクション数 200 から 600 であり，ダウンロード速度が向上した．

しかし，この評価では，単位時間あたりのトランザクション数が 800 程度になるとキャッシュ

の効果が得られなくなる．

図 5.3 にキャッシュ有無による性能比を示す．同一要求のもとで，キャッシュ有無による速度比を示している．アクセス数のある特定コンテンツでは，キャッシングした場合のダウンロード速度比がトランザクション数が多いほど高くなる傾向があり，負荷が増加するほど有効に機能する．

図 5.4 に対象コンテンツと一般コンテンツの性能比を示す．loadBlocks は 1 度にキャッシュするブロック数である．loadBlocks，numAccess によらずトランザクション数の増加にともない，性能比が向上する結果が得られた．アクセス数のある特定コンテンツでは，キャッシングした場合のダウンロード速度比がトランザクション数が多いほど高くなっており，負荷が増加するほど有効に機能する．

図 5.5 に対象コンテンツのキャッシュヒット率を示す．対象とするコンテンツのキャッシュヒット率は，50 から 80%であった．トランザクション数が増えるとヒット率が上がっているのは，ランダムに生成されるトランザクションに対象が含まれているため，トランザクション数がふえると numAccess によりキャッシュに残っている場合があるためである．

簡単な方法による固定長ブロック単位のキャッシングにより，コンテンツのアクセスが集中した際に，ダウンロード速度の改善およびトラヒック軽減ができることをシミュレーションで確認した．

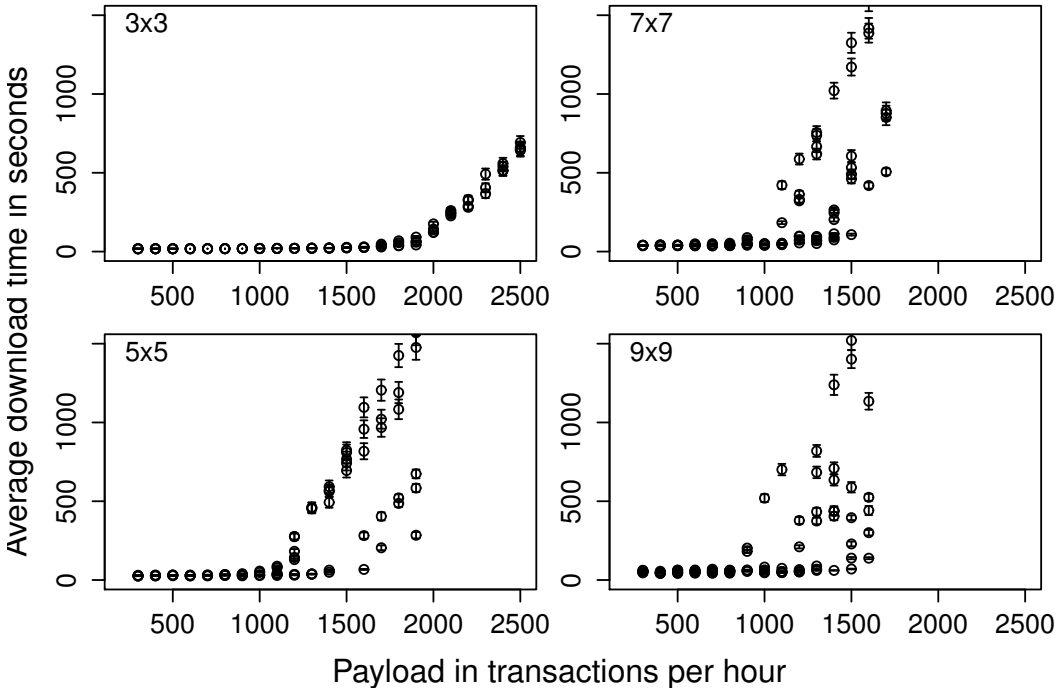


図 5.6: 格子状構成：平均ダウンロード時間（格子サイズによる影響）

5.1.2 格子状ネットワークモデルでの性能評価

このシミュレーション結果は，二つの部分から成る．一つは，エンドツーエンド通信（Traditional 方式）の予備的な性能評価結果で，もう一つは，Traditional 方式と提案する TCR アーキテクチャ方式の性能比較である．

格子構成：格子サイズの影響

格子サイズが 3×3, 5×5, 7×7, 9×9 の評価ネットワークで，時間あたりのダウンロード数を増加させた場合の平均ダウンロード時間とそのときの平均通信経路長を測定する．コンテンツの検索は URI により行う．経路上でのキャッシュは行わない．

図 5.6 に異なるネットワーク規模による平均ダウンロード時間を示す．ネットワークの規模が大きくなるにつれて，平均ダウンロード時間も大きくなるが．3×3 の格子サイズのダウンロード時間は，単純な M/M/1 待ち行列モデル [108] に従っており，図 5.6 に表示している 95% 信頼区間（プロットの上下横線）から変化は小さいことがわかる．格子サイズが増えるにつれて平均ダウンロード時間が急増する負荷が小さくなり，結果のばらつきも増加する．

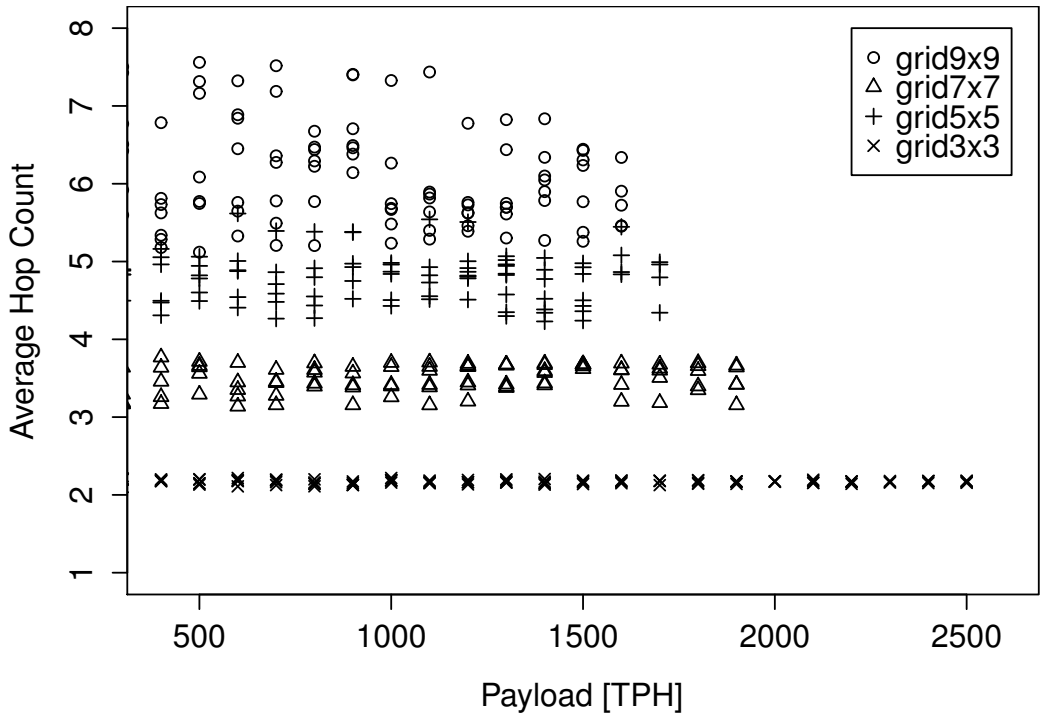


図 5.7: 格子状構成：平均経路長（格子サイズによる影響）

図 5.7 は，図 5.6 を測定した際の通信経路長である．3×3 では安定している経路長が，規模を大きくするにつれて大きく不安定になる．

以上から負荷の増加により結果が安定的でなくなる格子サイズは 5×5 であるため，次節では，提案する TCR アーキテクチャと Traditional 方式の平均ダウンロード時間を格子サイズ 5×5 で比較する．

格子状構成：性能試験の結果と解析

5×5 の格子サイズの評価ネットワークで，キャッシュを持たない伝統的なエンドツーエンド通信（Traditional 方式）とキャッシュを有する TCR アーキテクチャを比較する．

コンテンツはランダムに内部ルータに割り当てられる．TCR アーキテクチャでは，各コンテンツの断片化されたチャンク単位で内部ルータ上にキャッシュされる．各 TCR は状況に応じてコンテンツの探索とキャッシングを行う．

Traditional 方式では，各コンテンツは，CMS 機能のある内部ルータのオリジナルコンテンツを取得するために最短の経路でルーティングする．

TCR アーキテクチャでは，全検索中でタグを使用する割合であるタグ検索率 $tagsearch\%$ を変

えダウンロード時間の影響を確認する．タグ検索率 $tagsearch\%$ と URI 検索率 $URISearch\%$ の関係を式 5.1 に示す．

$$tagsearch\% \equiv (100 - URISearch\%) \tag{5.1}$$

tagsearch%=100 では検索は全てタグを用い，tagsearch%=0 では全ての検索をキーワードで行なうことを意味する．

図 5.8 にペイロード（トラヒック負荷）対平均ダウンロード時間を，図 5.9 に図 5.8 を測定したときの平均ホップ数を示す．

図中凡例の NoCache がエンドツーエンド（Traditional 方式）を，tagsearch%がタグ検索率を意味する．エンドツーエンドでは URI の検索のみ対応しているため $tagsearch\% = 0$ である．

図 5.8 のように，すべての $tagsearch\%$ において，TCR アーキテクチャでは常にエンドツーエンドに対して優位である．TCR アーキテクチャの $tagsearch\%$ が増加するにつれて，平均ダウンロード時間は短くなり性能は向上した．

図 5.9 では，キャッシュを用いない場合の平均経路長は 3.5 程度である．キャッシュを用いた場合でもタグ検索率が 0%では，キャッシュを用いない場合と同程度の平均経路長であった．タグ検索率 $tagsearch\%$ を増やすにつれ平均経路長は短縮され，$tagsearch\%$=80 では，$tagsearch\%$=0 の約半分まで低減できた．

以上から，キャッシュを用いると平均ダウンロード時間に効果があり，タグ検索率は，平均経路長低減に効果がある．

アクセス能力の向上の程度を推定するために，CGM コンテンツ視聴シナリオを考える．

動画の再生は，データ断片を再生している間，次のデータ断片のダウンロード完了しなければならない制約がある．ダウンロード速度が再生速度がより低下すると視聴が中断する．例えば 10 分（600 秒）の再生時間を持ちサイズが 100M バイトのビデオファイルの視聴を行う場合を考える．

図 5.8 では，エンドツーエンドでは，約 1500TPH（時間あたりのトランザクション数）までのネットワーク全体の負荷を許容できる．

一方，TCR アーキテクチャでは，URI ベースのみの検索（$tagsearch\% = 0$）では，約 9,000TPH まで対応でき，従来のアーキテクチャに比べて 6 倍優れている．さらに，タグによる検索が 80%の場合は，17 倍の性能を達成し，約 26,000TPH を処理することができる．

図 5.10 に経路長とヒット率の関係を示す．横軸は，視聴端末からの平均経路長で，縦軸は，出現率である．キャッシュを用いないエンドツーエンド（NoCache）では，経路長 3 と 4 の間をピークとする分布が得られた．タグ検索率（tagsearch%）を増やすにつれ，ピークは平均経路長が小さい方へ移動し，タグ検索率 100%では経路長 1 と 2 でデータを取得できている．

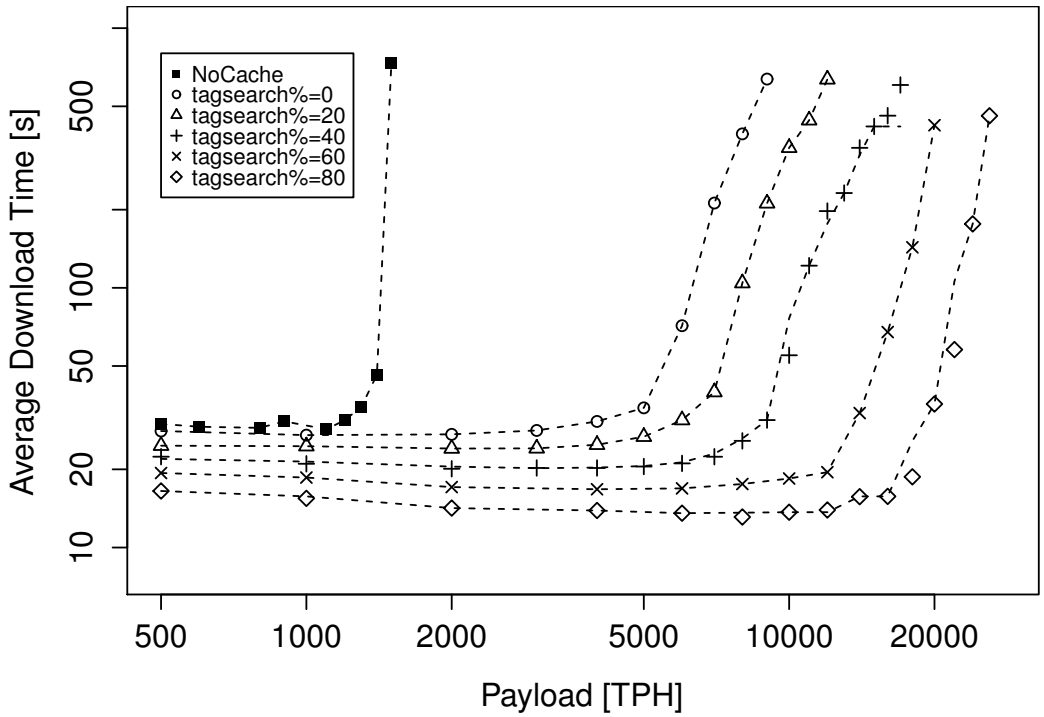


図 5.8: 格子状構成：平均ダウンロード時間（タグ検索率による影響）

理想的な CDN の場合は，経路長 1 で出現率が 100%となるが，ストレージコストの問題がある．経済性も含めた評価は総合評価で後述する．

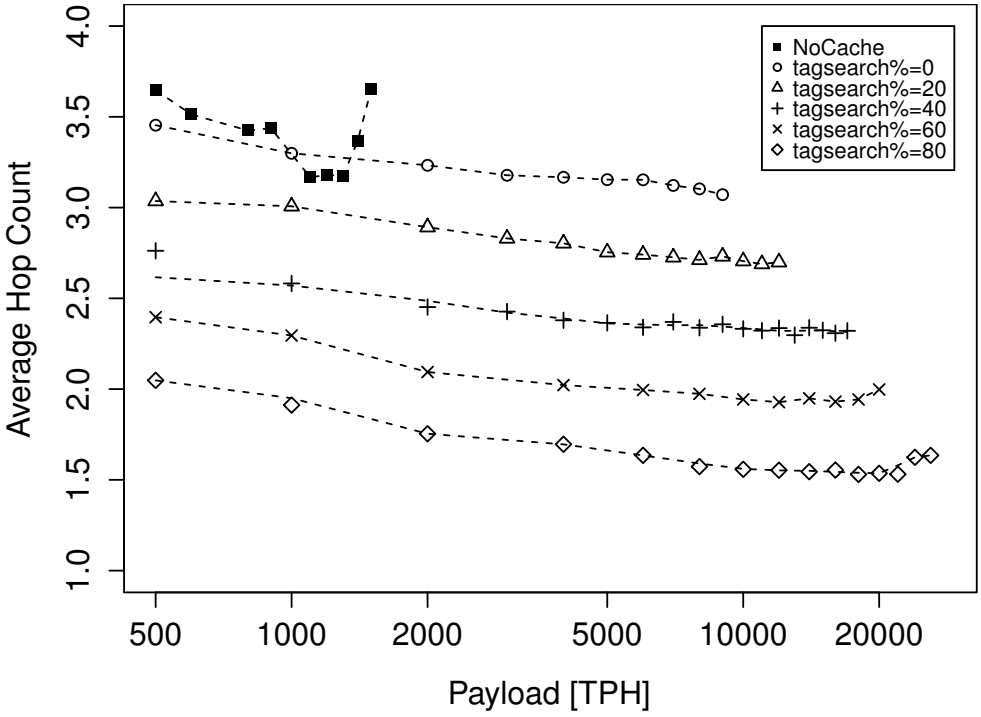


図 5.9: 格子状構成：平均経路長（タグ検索率による影響）

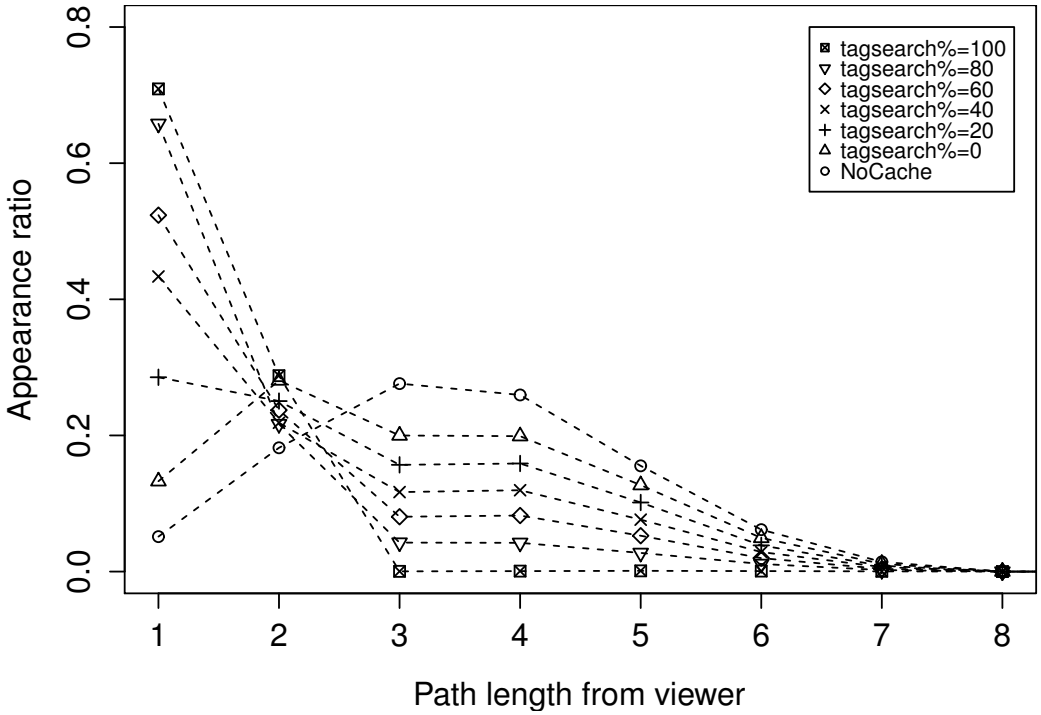


図 5.10: 格子状構成：ヒット率

5.1.3 ツリー状ネットワーク構成での性能評価

国内にある CMS サーバから CGM 動画視聴を行う場合を想定して，平均経路長が 7 程度となる BA ネットワークでタグによるキャッシングの性能を評価する．

初めに，結果サマリを提示し，次に，主要なシミュレーションパラメータであるタグ数，割引率，キャッシュ容量比の影響を，最後に，データを取得した経路長について確認する．

結果サマリ

TCR のテストケースの中で最良ケースと最悪ケース，URI によるキャッシュ方式，キャッシュを用いない方式の比較を行う．提案する TCR アーキテクチャの結果は，”Tag_”を接頭子で提示する．

ダウンロード要求に対する応答性の関係図 5.11 から 5.14 に示す．横軸はいずれも時間あたりのダウンロード要求数 $nreqs/hour$ でスケールは 10^3 である．

平均ダウンロード時間を図 5.11 に，その標準偏差を図 5.12 に，その時の平均経路長を図 5.13 に，経路長の標準偏差を図 5.14 に示す．図 5.11, 図 5.12 の縦軸は，対数である．いずれのグラフも縦軸の値が少ないほど性能が優れていることを表している．Tag-based では今回用いたシミュレーション条件で，最良と最悪の結果をプロットしている．凡例で URI_100 は，URI-based でキャッシュ容量比 $cache\%$ が 100%を意味する．Tag_1_0_12 は，Tag-based でタグ数 $ntag$ が 1，割引率 d が 0%，キャッシュ容量比 $cache\%$ が 12%を意味する．以下同じ意味として用いる．点線 Ideal Value は，端末が接続しているエッジのルータからデータを取得し，端末間の競合が無い場合の CDN での理想値に相当する．

図 5.11 のダウンロード時間で，Traditional では，$nreqs/hour$ の増加により性能が悪化したが，URI-based および提案する Tag-based では，安定している．Tag-based の最良ケース Tag_1_0_12 では，URI_100 より優れているが，最悪ケース Tag_8_6_04 では，劣った．図 5.12 で，Traditional では，$nreqs/hour$ の増加に伴い悪化するが，URI-based，Tag-based とも図 5.13 で経路長が短いことから安定している．図 5.13 の平均経路長では，$nreqs/hour$ の増加により URI_100 は改善しているが，他は一定であった．図 5.14 では，いずれも $nreqs/hour$ の増加により減少した．

タグ数 $ntag$ または割引率 d の値により Tag-based は，URI-based に対して優劣があるが，平均ダウンロード時間は同程度であり，キャッシュ容量が URI-based に対して 4 から 12%でもダウンロード時間や経路長を短くできることを確認した．

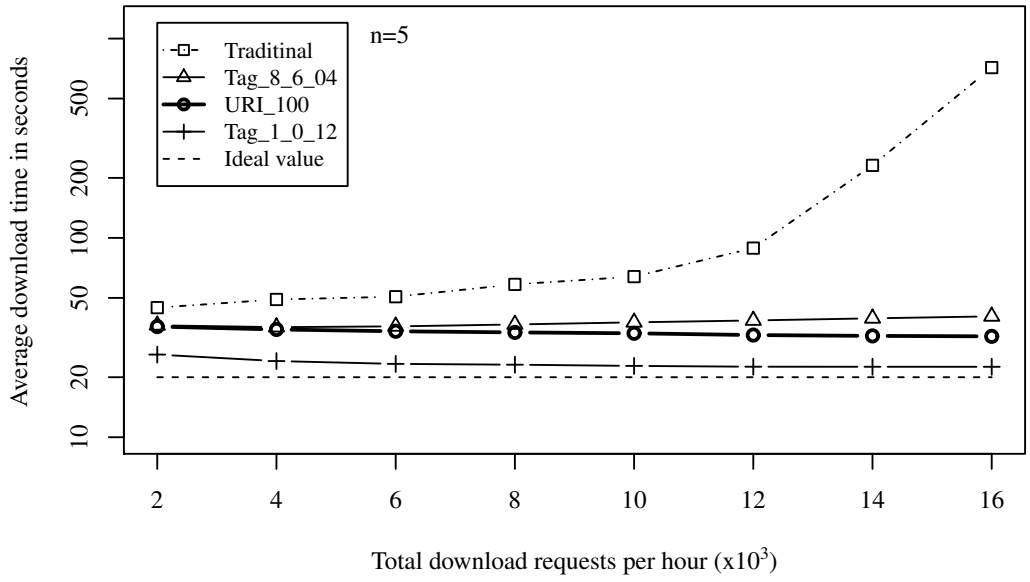


図 5.11: ツリー状構成：平均ダウンロード時間

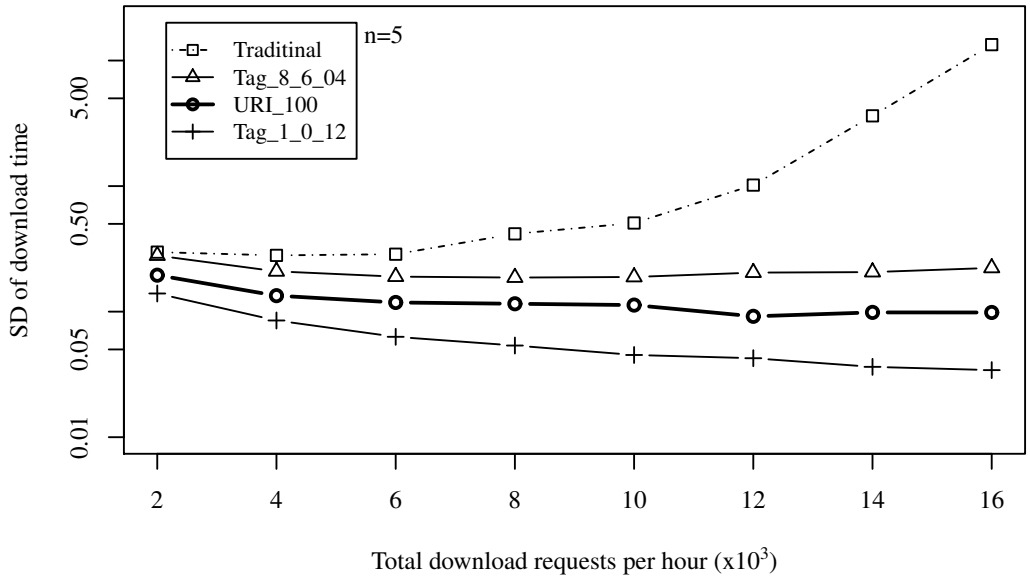


図 5.12: ツリー状構成：平均ダウンロード時間の標準偏差

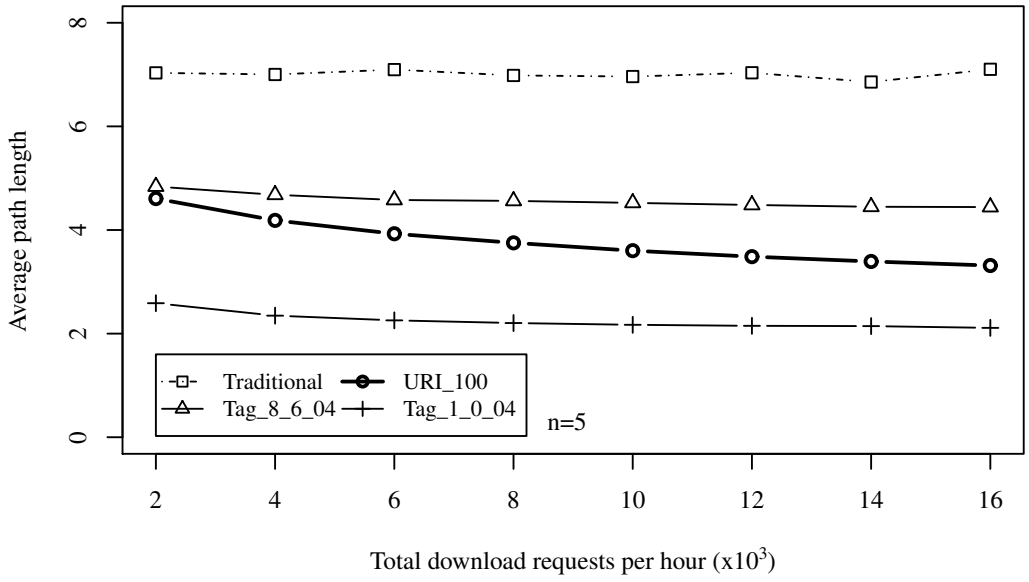


図 5.13: ツリー状構成：平均経路長

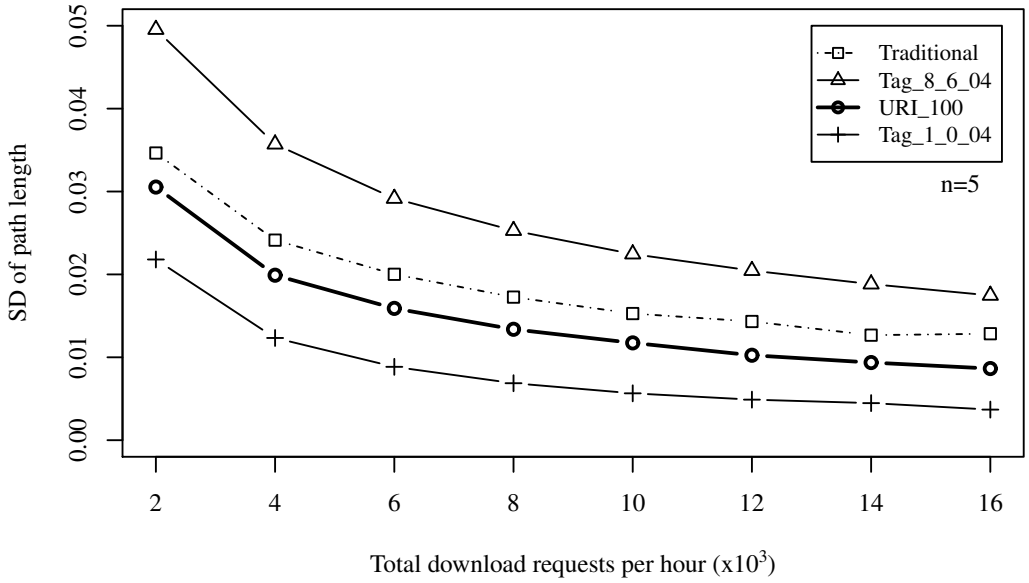


図 5.14: ツリー状構成：平均経路長の標準偏差

ツリー状構成のタグ数，割引率，キャッシュ容量比による影響

図 5.15 から図 5.18 にタグ数 $ntag$，割引率 d, または，キャッシュ容量比 $cache\%$ を変えた場合の平均ダウンロード時間を示す．

横軸は図 5.11 と同様である．比較のため URI_100 を太線で表している．タグ数 $ntag$=1 の場合が図 5.15 および図 5.17 で，$ntag$=8 の場合が図 5.16 および図 5.18 である．図 5.15, 図 5.16 では，キャッシュ容量比 $cache\%$ が，図 5.17, 図 5.18 では，割引率 d がパラメータである．図 5.15, 図 5.16 から割引率 d が大きくても $cache\% > 4\%$ では，性能は URI_100 より優位である．図 5.17, 図 5.18 では，割引率 d の増加により性能は悪化するが，ダウンロード時間の変動率は，最大の Tag_8_6_04 でも 14%であった．d が同じである図 5.16 Tag_8_6_12 では，性能が改善されており，割引率 d=6% では各タグの生起確率が同程度であることから，TCR 方式では，キャッシュ容量が少なくてもロングテールの変動に対応できる．

$ntag$=$2or4$ での結果は，最悪ケースと最良ケースの間となったため図は省略する．

キャッシュヒットに影響を与えるタグ数や割引率は，応用によって決まる値なので，TCR アーキテクチャには，キャッシュヒットの状況からキャッシュポリシを自動調整する機能が求められる．

ツリー状構成の経路長

累積ヒット率分布を図 5.19 から図 5.22 に示す．

縦軸は，出現数の累積値で，集計対象の取得先は，図 5.19, 図 5.20 では CMS を含み，図 5.21, 図 5.22 では CMS を含まずルータのみである．横軸は，ダウンロード要求を発行した端末からの経路長である．

図 5.19, 図 5.20 では，一点鎖線で示した Traditional は，BA ネットワークトポロジーでは正規分布状となった．URI-based, Tag-based ともに同様の傾向となるが $ntag$=1 の場合は，d が少ないほど立ち上がりが緩やかとなった．しかし，図 5.21, 図 5.22 のキャッシュヒット率を見ると d が低いほど，短い経路長でのキャッシュ使用が高く，上位ランクのタグを持つコンテンツがキャッシュされている．特に図 5.21 で d=0%の場合は，経路長が 1 でヒット率 100%となる理想的な CDN に近い結果を得た．Traditional の平均経路長である 7.2 付近でキャッシュヒット率は飽和した．図 5.22 の最悪ケース Tag_8_6_4 では $cache\%$=4 かつ各タグの出現頻度が同程度の条件下でも経路長 3 で 50%のヒット率を得た．

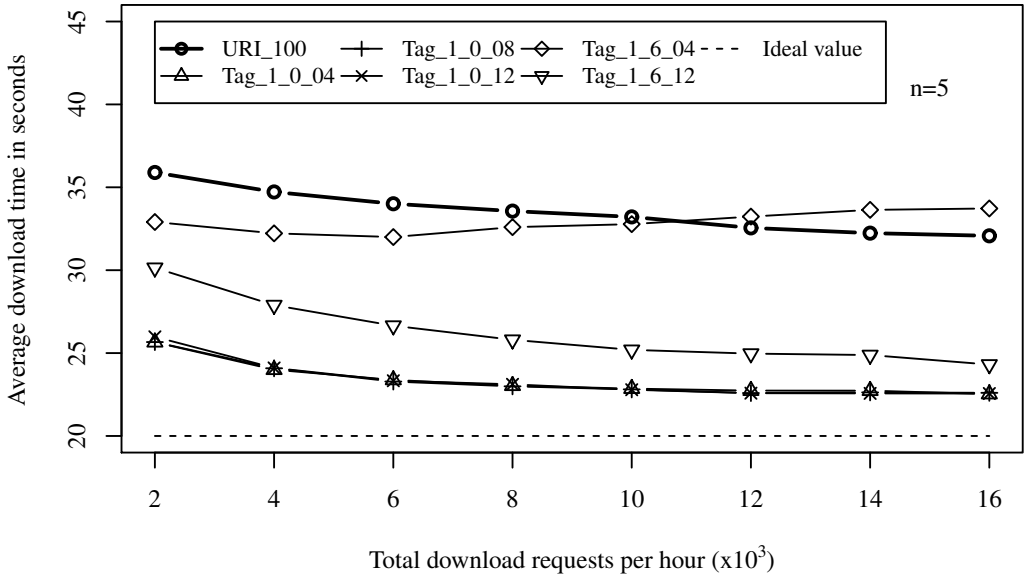


図 5.15: ツリー状構成：平均ダウンロード時間（$ntag$=1）

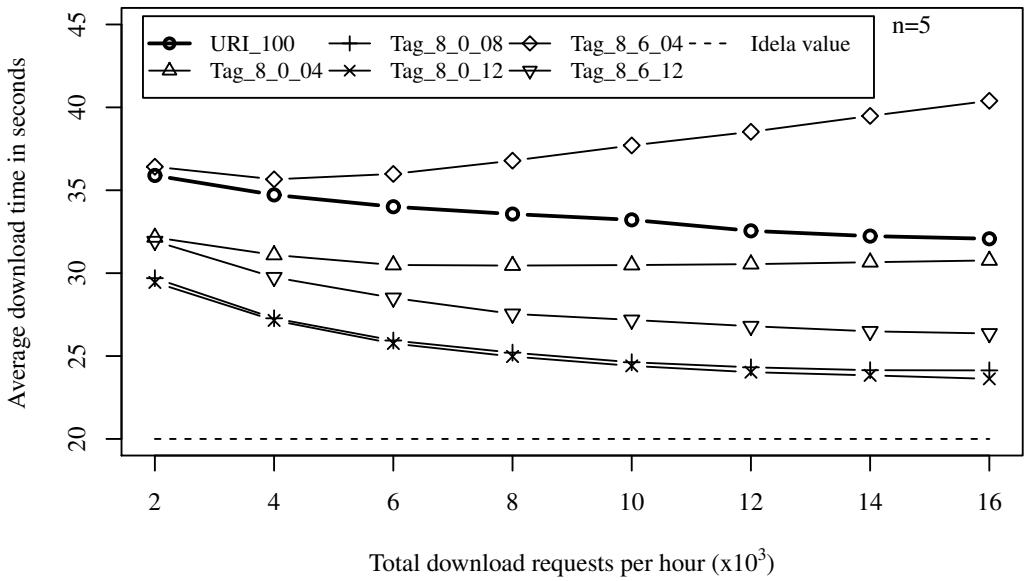


図 5.16: ツリー状構成：平均ダウンロード時間（$ntag$=8）

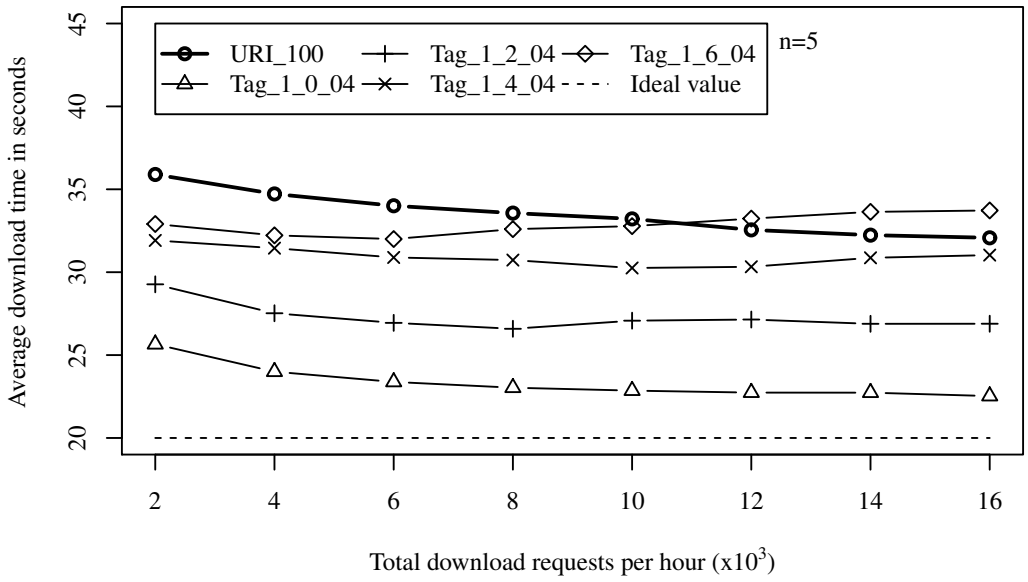


図 5.17: ツリー状構成：平均ダウンロード時間の割引率による影響（$ntag$=1）

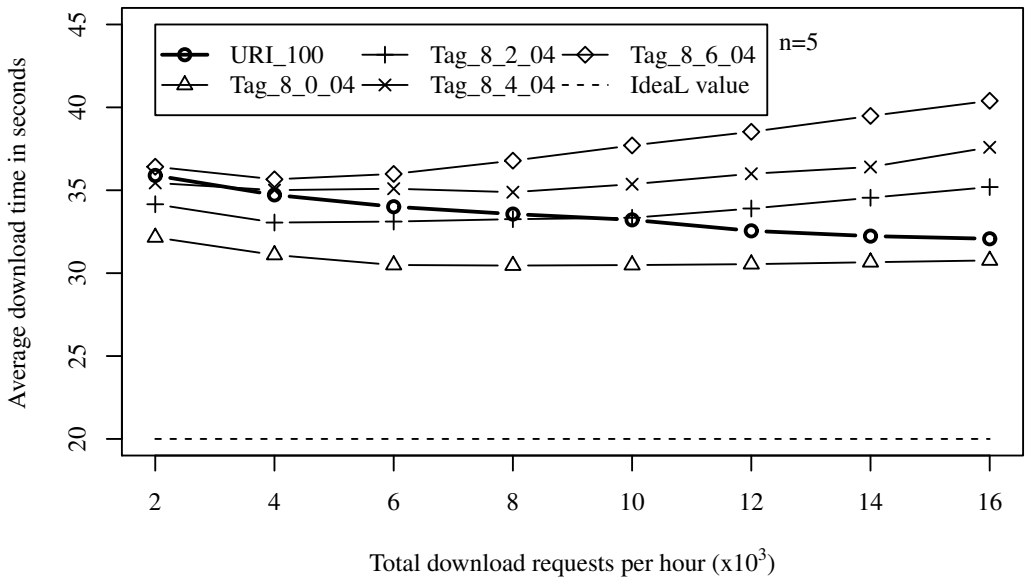


図 5.18: ツリー状構成：平均ダウンロード時間割引率による影響（$ntag$=8）

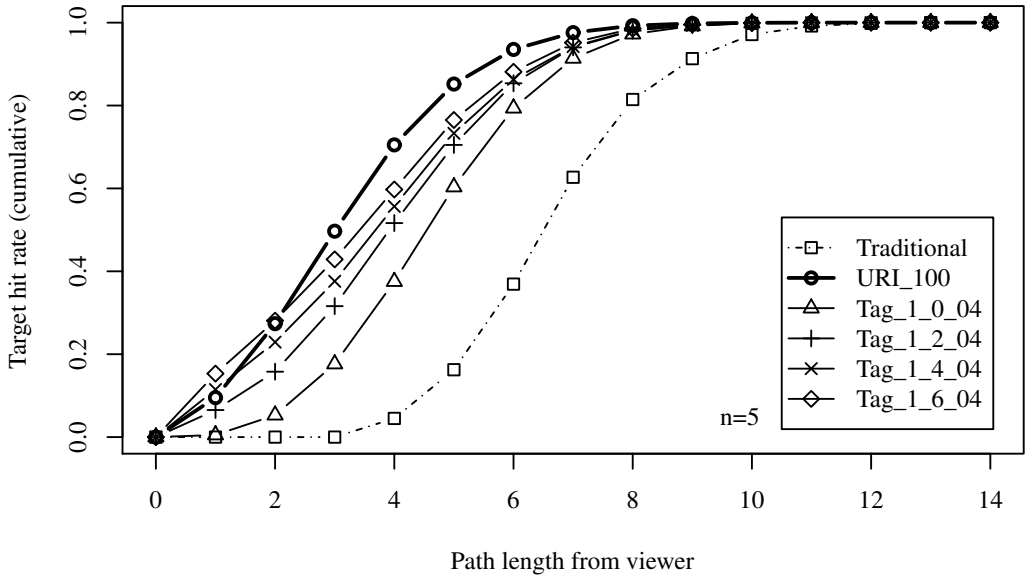


図 5.19: ツリー状構成：データ取得位置（$ntag$=1, $nreqs/hour$=16k）

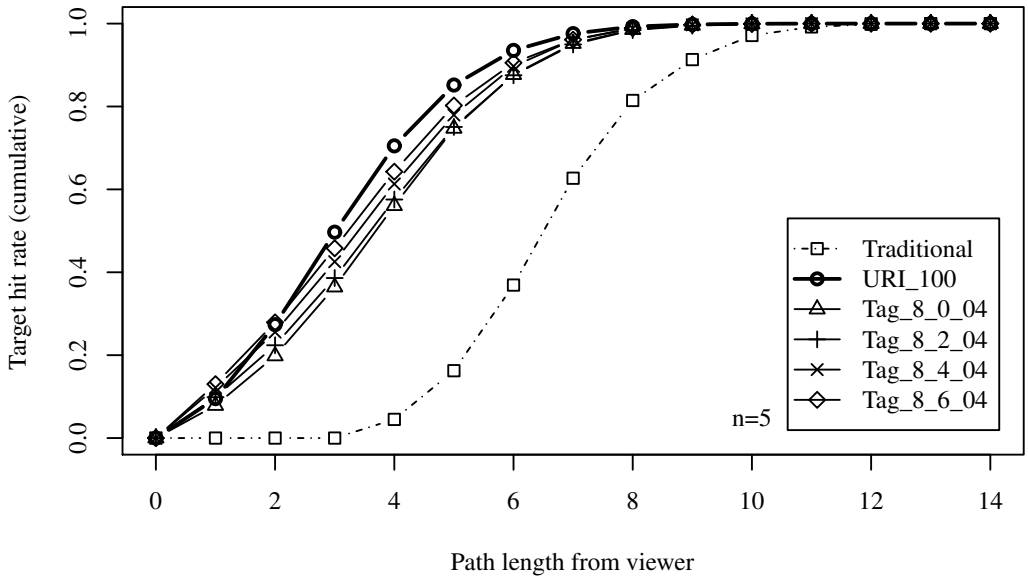


図 5.20: ツリー状構成：データ取得位置（$ntag$=8, $nreqs/hour$=16k）

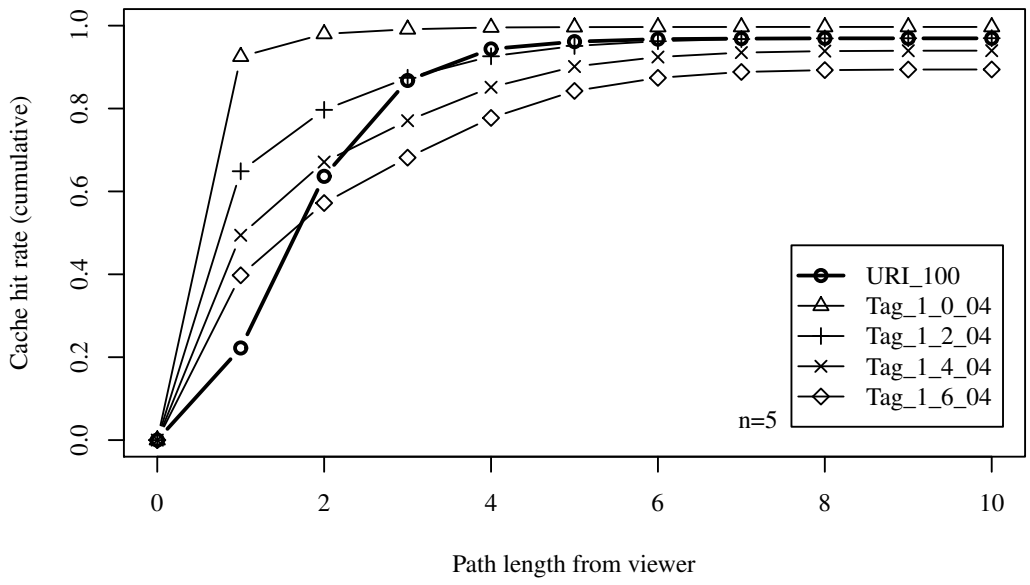


図 5.21: ツリー状構成：キャッシュヒット率（$ntag$=1, $nreqs/hour$=16k）

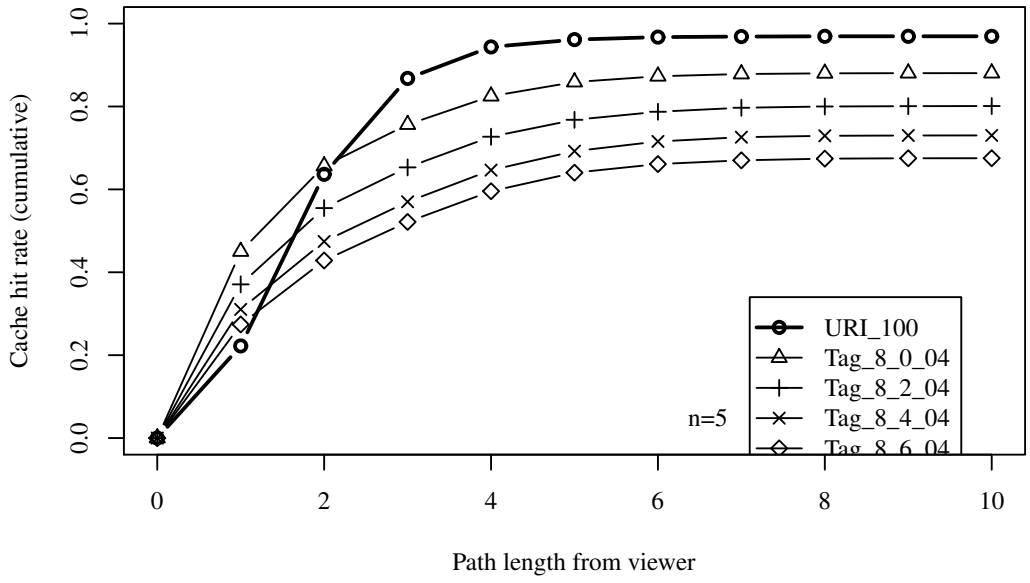


図 5.22: ツリー状構成：キャッシュヒット率（$ntag$=8, $nreqs/hor$=16k）

まとめ

平均経路長が 7 程度となる BA ネットワークでタグによる経路上のキャッシュは，タグ分布がロングテール状となることを利用すると負荷を増加させた場合に性能の維持を図ることができる．ただし，ロングテールの裾野が長くなるとキャッシュヒット率が落ち，性能向上の度合いが低下する．また，コンテンツに割り当てるタグ数が多くなると同様な傾向が見られる．

5.2 階層キャッシュ方式による影響

ツリー状構成の評価ネットワークと評価条件を用いてキャッシュ登録方法，キャッシュ廃棄方法の影響を確認する．評価条件は 5.1.3 節と同様であるが，表 4.3 のキャッシュ登録方式（Entry method）とキャッシュ廃棄方式（Discard method）を変更する．

登録方法に LCE, LCD, MCD，廃棄方法に LRU, FIFO があり，登録と廃棄の組み合せは 6 種類となるが，登録方法の LCD は LCE のサブセットであり，廃棄方法の FIFO はアクセス状況を反映しないので評価から除外する．

平均ダウンロード時間を図 5.23 に，平均ダウンロード時間の標準偏差を図 5.24 に，平均経路長を図 5.25 に，平均経路長の標準偏差を図 5.26 に，視聴端末からの取得位置を図 5.27 に，キャッシュヒット率を図 5.28 に示す．

凡例の意味を記す．Ideal Value は論理的最速の値である．Traditional はキャッシュを用いないエンドツーエンドアーキテクチャである．URI で始まるものは，キャッシュはするがキーは URI を使用する．URI_4 の 4 はキャッシュ容量比 $cache\%$ でキャッシュサイズがコンテンツデータ総量の 4%を意味する．Tag で始まるものは，タグをキーとしてキャッシュする TCR アーキテクチャである．続く数字は，コンテンツに割り当てるタグの数 $ntag$，割引率 d，キャッシュ容量比 $cache\%$ である．

図 5.23 では，URI によるキャッシュの性能は，キャッシュしない場合と同様となった．これは，$cache\%$ を大きくすると，CDN と同じように性能を改善できるが，キャッシュストレージの経済性のトレードオフが生じる．

タグをキーとした TCR では負荷の増加に対して性能の悪化は計測範囲内では生じなかった．

いずれのアーキテクチャでも LCE と MCD の性能の相違は無かった．

図 5.24 は図 5.23 を計測した際の平均ダウンロード時間の標準偏差で，縦軸は対数表示である．結果は，図 5.23 と同様で，LCE と MCD の顕著な差は見られなかった．

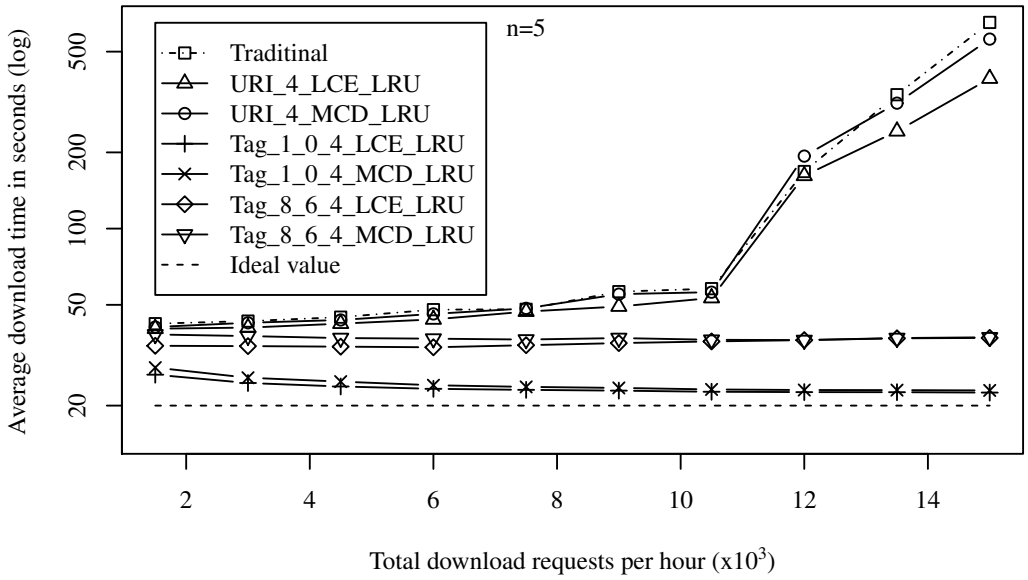


図 5.23: キャッシュ登録方式：平均ダウンロード時間

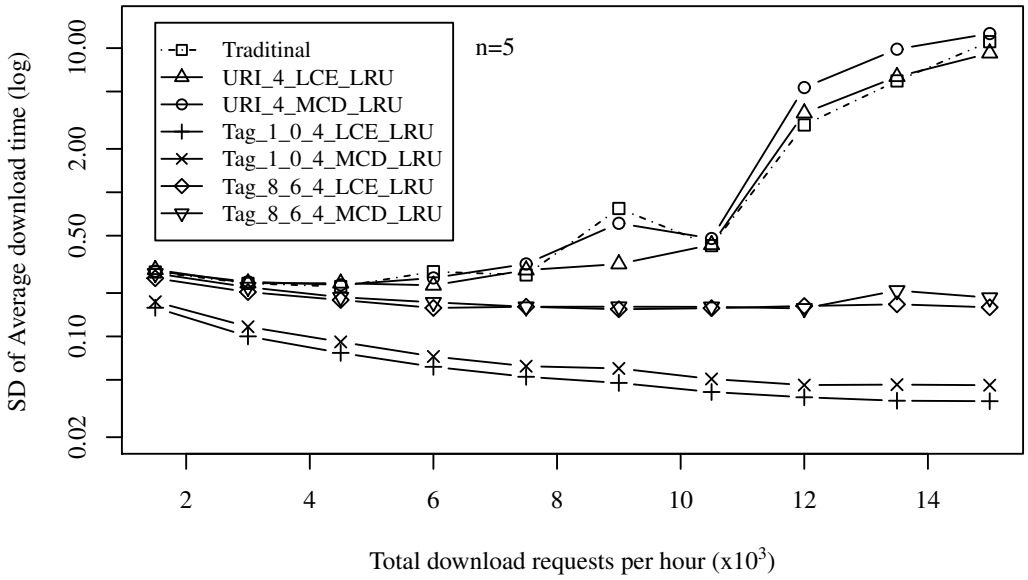


図 5.24: キャッシュ登録方式：平均ダウンロード時間の標準偏差

図 5.25 の平均経路長は，方式により同様の値となり，LCE，MCD の相違は見られなかった．

図 5.26 は，平均経路長の標準偏差であるが，いずれも負荷が増えると標準偏差が低くなる．特に，コンテンツに割り当てるタグ数 $ntag$ が 1 のときに低減の割合が大きい．図 5.27 ではデータを取得した端末からの経路長に対して出現頻度を累積表示している．キャッシュを用いない Traditional は平均経路長が 7 の付近を対称とする正規分布的な傾向がある．URI によるキャッシュでは，端末よりの出現頻度が高い．しかし図 5.23 で性能が Tag をキーとしている場合より悪いことから，エッジルータやエッジに近い位置のルータの待ち行列の長さが長くなっていると考えられる．

図 5.28 はキャッシュ置換率（キャッシュ廃棄回数／ダウンロード要求数）を端末からの経路長毎に表示している．タグをキーとした場合，タグ割当数 $ntag$ が 1 の場合は，キャッシュ置換が発生していない．ついで，$ntag = 8$ の場合のキャッシュ置換率が生じている．LCE では，経路上全てのノードのキャッシュを更新するため，MCD に比べてキャッシュ置換率が高くなっている．URI は，キャッシュ置換が多発しており，特にエッジ側のキャッシュ置換が多い．

以上から，URI でのキャッシュ方式は，キャッシュ容量が小さいとネットワーク負荷の増加に対して，性能の改善が図れず，かつエッジ側のルータでキャッシュ置換が多発する．キャッシュ容量が十分ある場合は，登録方式が LCE でも MCD でも性能，通信経路長の差は少ない．

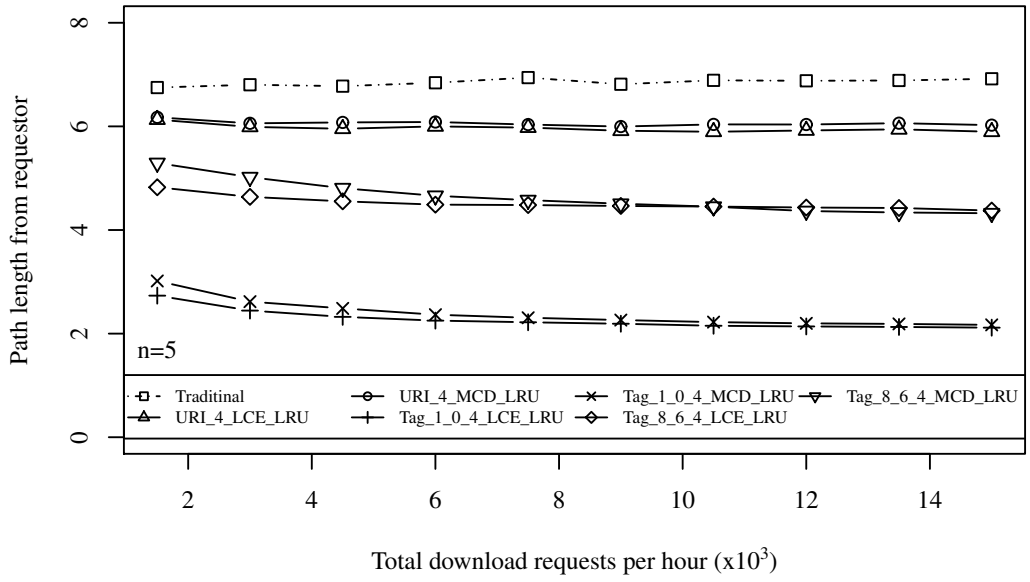


図 5.25: キャッシュ登録方式：平均経路長

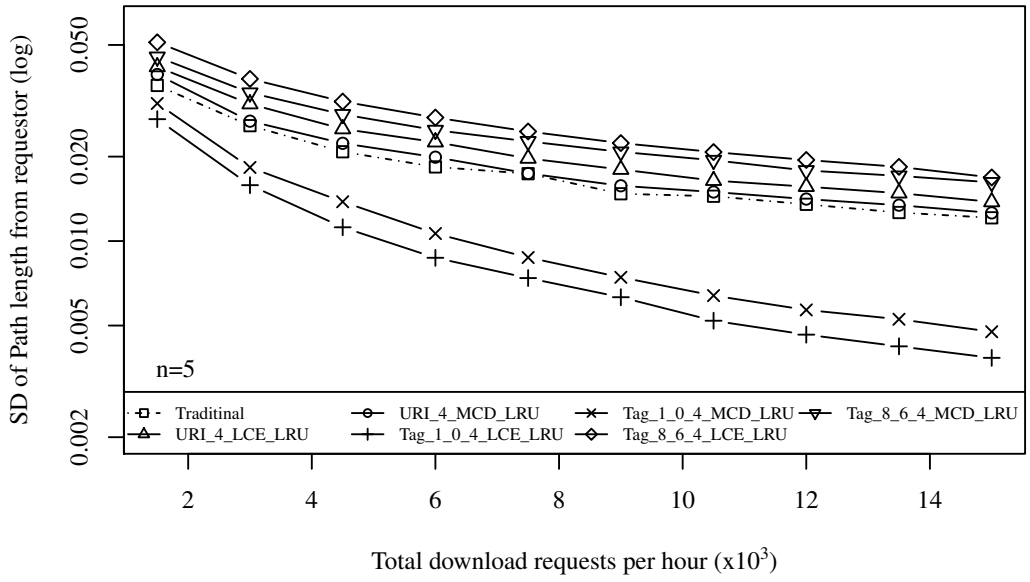


図 5.26: キャッシュ登録方式：平均経路長の標準偏差

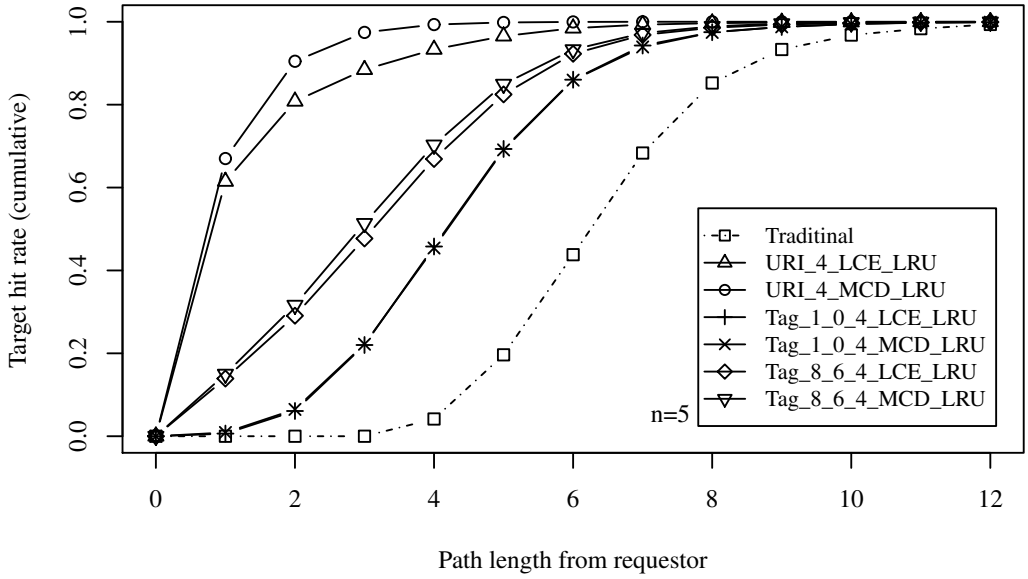


図 5.27: キャッシュ登録方式：データ取得位置

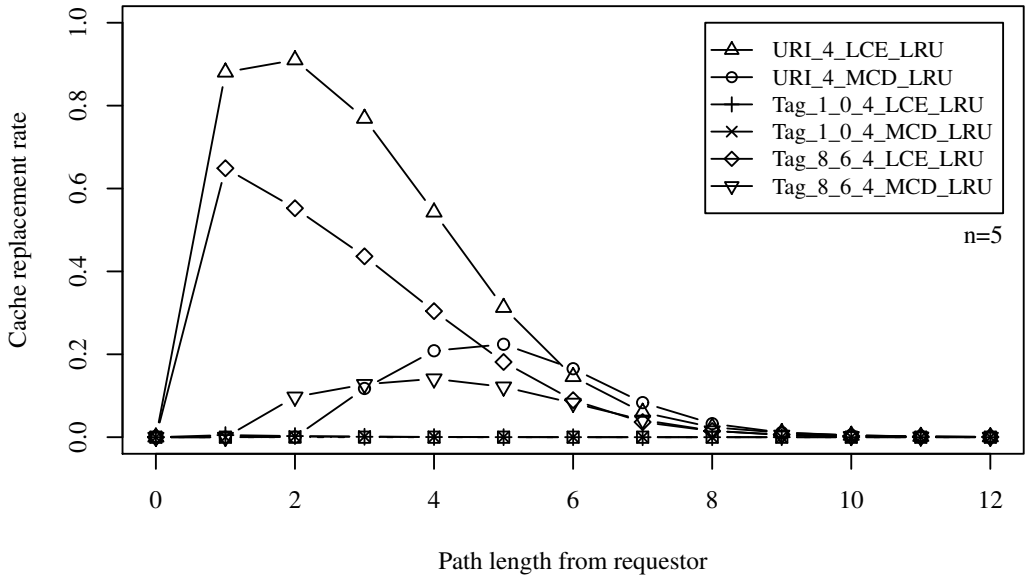


図 5.28: キャッシュ登録方式：キャッシュ置換率

5.3 総合評価

総合評価の結果を図 5.29 に示す．時間当たりのダウンロード要求数 $nreqs/hour$ の増加に伴い，Traditional では，スコアが低下するが，経路上にキャッシュする URI-based および Tag-based の方式では安定している．URI-Based の総合評価は Tag-based より 10〜100 倍劣っている．Tag_1_0_12 ($ntag$=1, d=0%, $cache$%=12) の総合評価値は，理想状態の 1/100 程度であり，比較した方式の中では 1 番良い結果である．また，負荷の増加に伴い総合評価値が僅かであるが改善した．

Traditional 方式は, 負荷が少ない場合に総合評価値が高いが, $nreqs/hour$=6000 付近で, Tag_8_6_04 ($ntag$=8, d=6%, $cache$%=4) より，$nreqs/hour$=12000 付近で，URI_100($cache$%=100) より劣位となった．

以上から Tag による経路上にキャッシュする方式が総合的に優れており，負荷の増加に対する影響も少ない．

総合評価の内訳について応答性，安定性および経済性の各項目ごとに確認する．

図 5.30 に応答性の比較を示す．動画ダウンロードの応答性は，平均ダウンロード時間により評価する．ここで基準値は，CDN で回線の競合が起こらない理想のダウンロード時間である．時間あたりのダウンロード要求数 $nreqs/hour$ の増加に伴い Traditional 方式が悪化しているが，経路上にキャッシュを持つ URI-based および Tag-based 方式では負荷の影響を受けていない．

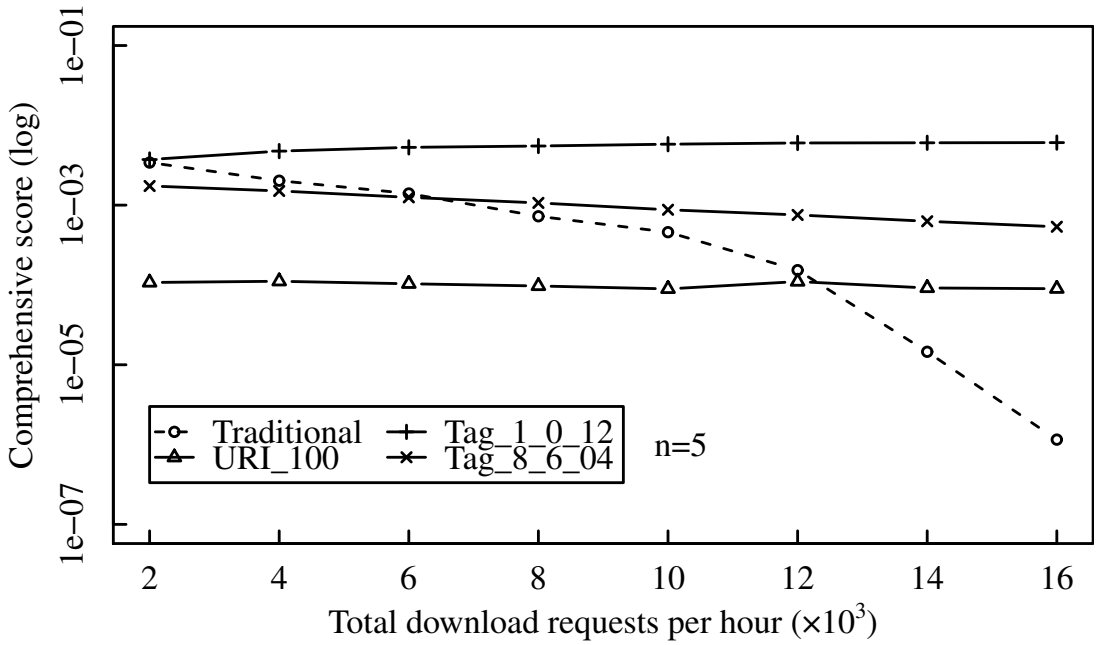


図 5.29: 総合評価：結果

URI-based および Tag-based 方式のダウンロード時間は，CDN での理想値に比べ 1/2 以内である．

図 5.31 に安定性の比較を示す．ここで基準値は，各方式のうちダウンロード時間の標準偏差が最小のものを用いる．Tag_1_0_012 では，ダウンロード時間の標準偏差が常に最小のため，相対値は時間あたりのダウンロード要求数 $nreqs/hour$ によらず 1 である．負荷の増加に伴い Tag_1_0_012 以外の相対値は悪化するが, 特に Traditional 方式が顕著である．URI_100 の安定性相対値は, Tag_1_0_012 と Tag_8_6_004 の間の値であった．

Traditional の安定性相対値は，負荷によらず他方式より劣位である．これは，通信経路長が大きく，ルータでの回線待ち時間が累積する場合が増えるためと考える．

経済性は，ストレージ経済性と使用帯域経済性の積からなる．

図 5.32 にストレージ経済性を示す．基準値は，各方式のうち最小のストレージ容量をもつ方式で, ストレージは, コンテンツを永続的に保存するディスクストレージとルータ上のコンテンツデータのキャッシュの総容量である．ルータにコンテンツデータ用のキャッシュを持たない Traditional 方式がストレージ容量が一番小さいため，この値が基準値である．

ここでは時間あたりのダウンロード要求数 $nreqs/hour$ の増加に伴い各方式のスコアは同等である．URI-based 方式および Tag-Based 方式でキャッシュは，すべて使用されているために方式によりストレージ経済性の値は．一定である．キャッシュ容量の大きな URI-based 方式が不利で, Traditional 方式のディスクストレージ容量を各ルータにキャッシュするため，ルータ数から 1/100 の値である．

図 5.33 に使用帯域経済性を示す．基準値は各方式で最小の使用通信帯域のものを採用する．評価ネットワークの回線速度は，表 4.3 より端末に接したルータとの間の回線速度は 100Mbps で，その他は 1Gbps であるから，使用帯域は 100Mbps/((平均経路長 $-$ 1) $*$ 1Gbps) により求めることができる．

ここでは，時間あたりのダウンロード要求数 $nreqs/hour$ の増加に伴い各方式のスコアは同等であった．Tag_1_0_012 が負荷によらず基準値であった．これは，通信経路長が短いことが寄与している．

以上から，負荷の増大に対して，Traditonal 方式は影響を受けやすく，URI-Based 方式は，多少悪化するが，Tag-based 方式は，影響を受けにくい．安定性は，どの方式でも負荷の増大に対して影響を受けるため，安定性の悪化を抑止することが総合評価を良くするために必要である．

安定性を高めるためには，通信経路長を短くし，かつコンテンツキャッシュを大きくすることが有効となる．CDN では，通信経路長が他方式より短く理想的であるが，タグ検索の対応，ストレージ容量および人気コンテンツ情報の反映間隔によっては，Tag-based 方式の選択または混在したコンテンツ配信基盤が考えられる．

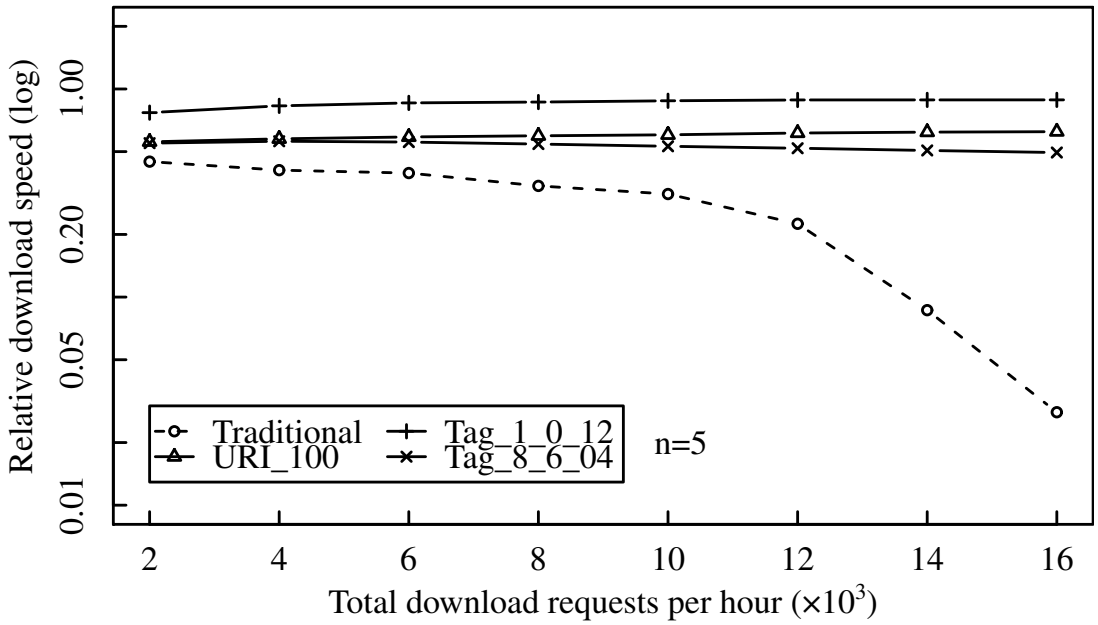


図 5.30: 総合評価：応答性の比較（CDN での理想値が基準値）

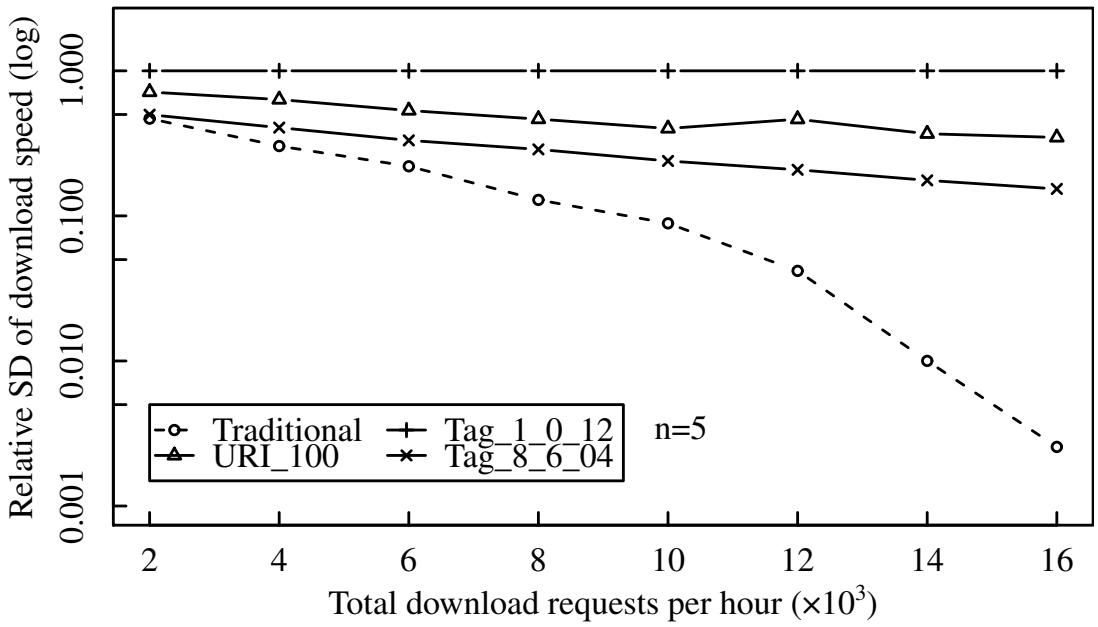


図 5.31: 総合評価：安定性の比較（各方式で標準偏差が最小の値が基準値）

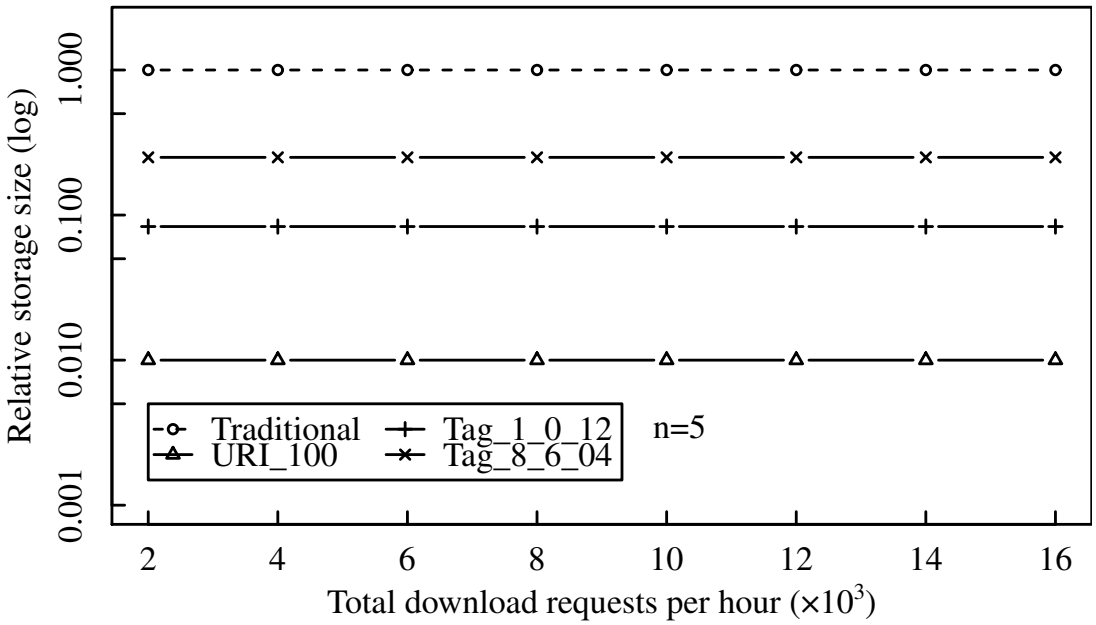


図 5.32: 総合評価：ストレージ経済性の比較（各方式で最小の値が基準値）

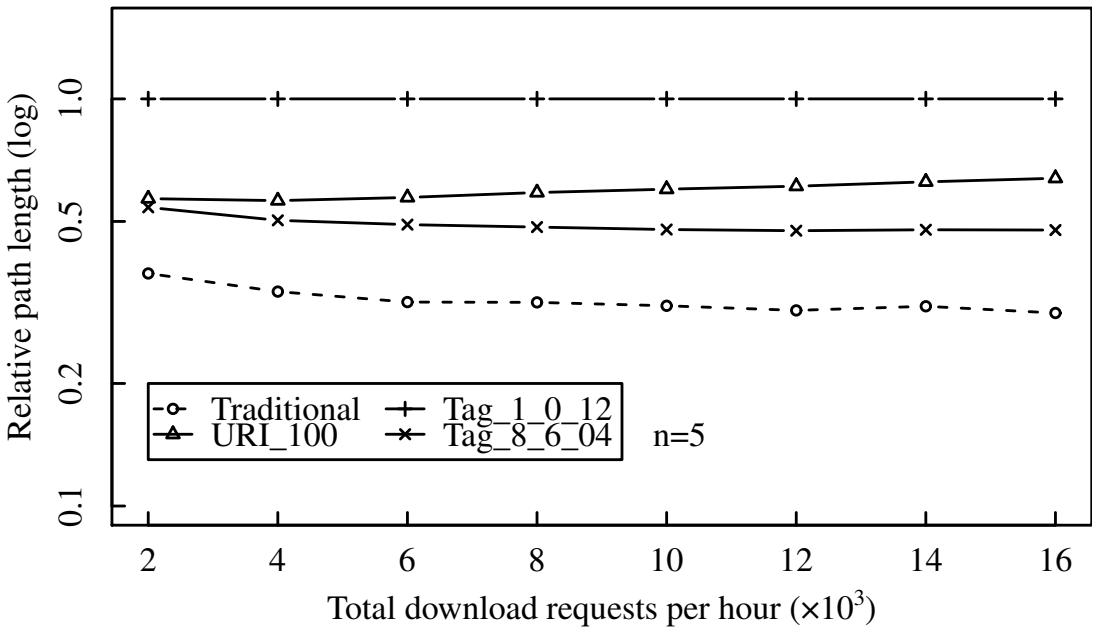


図 5.33: 総合評価：使帯域経済性の比較（各方式で最小の値が基準値）

5.4 まとめ

段階的にタグにより経路上のルータにキャッシュする TCR アーキテクチャを評価した．

初めに，5.1.1 節で経路上にキャッシュが 1 つ存在する基本的な構成で，アクセスが集中する場合にキャッシュの効果があることを確認した．

次に，5.1.2 節で平均経路長の差が生じにくい格子状トポロジーでキャッシュの効果を確認した．ここでは，タグ検索率による評価を行ない，平均ダウンロード時間はキャッシュルータを導入する事で負荷の増加に対する耐久性が増し，平均経路長は，タグ検索率を増加させることで改善の効果が高くなることを確認した．

5.2 節では，階層キャッシュの登録方式による影響より，キャッシュ容量が影響することを確認した．

5.1.3 節では，より現実的な動画視聴環境を想定し，タグランキング分布やコンテンツに割り当てるタグ数の影響を確認した．

5.3 節では，応答性，安定性，経済性による総合評価を実施した．

以上からタグによる経路上のキャッシュ方式は，URI によるキャッシュ方式より応答性，安定性および経済性に優れていると言える．

（意図的な空白ページ）

第6章　結論

本章では，提案した TCR（Tag Caching Architecture）アーキテクチャの妥当性を 6.1 節で，研究の有用性を 6.2 節で確認し，将来の展望を 6.3 で述べ，まとめる．

6.1　提案の妥当性

提案の妥当性をアーキテクチャと評価環境から確認する．

6.1.1　アーキテクチャの妥当性

FIA（Future Internet Architecture）では汎用性から以下を前提としている．

前提 1 コンテンツの名前解決とコンテンツデータ保持は別システムにより管理される

前提 2 コンテンツが任意の場所から生成され保持される

前提 3 コンテンツを一意に特定しコンテンツデータを取得する

これらの前提は，ソーシャルネットワークサービスにおいて必ずしも優位でないことをアーキテクチャのルーティングとキャッシングの面から述べる．

ルーティング

従来手法ではコンテンツの生成場所が任意の場所を暗黙の前提としている（前提 2）．そのため名前解決を行う特定のノードを設けるか，接続経路全てに照会する方法を取る．前者の例は，NDN（Named Data Network）[59] が，後者の例は Breadcrumbss [62] がある．

NDN の様に専用の名前解決ノードを持つ場合は，コンテンツ登録時，コンテンツ探索成功時およびキャッシング時に情報更新が行われる．

2.2 節で示したように CMS サイトの動画コンテンツタイトル数は膨大であり，視聴者が日々 CGM コンテンツを生成しているため，経路上のルータにキャッシュする場合にコンテンツ位置の更新要求が大量となり，かつ同一コンテンツを保持するルータ数が多くなることから，取得位置により有利なコンテンツ位置を返す名前解決機能が求められる．ここで前提 1 を除外すると Breadcrumbs や TCR の様に経路上のルータでの名前解決が考えられる．

一方，Breadcrumbs の様にコンテンツ探索の照会要求を経路上に送信する方法は，名前解決を行うコンテンツレゾルバが単純になるメリットがあるが，ルータが接続されている全てのリンクに照会要求を送信するため，探索を打ち切るタイムアウト値の設定・調整が必要となる．

ここで，SNS，動画投稿サイトなどはオリジナルデータを一元管理することから前提 2 を除外し，視聴者からオリジナルコンテンツを持つ CMS へ向けて照会要求を送信することで照会要求を削減でき，最終的にコンテンツを取得できる．またタイムアウト値の調整も不要となる．

以上からオリジナルの情報が集中する場合のコンテンツ探索のルーティング方法は提案する TCR が優れている．

キャッシング

従来手法ではコンテンツを通信経路上にキャッシュする場合に前提 3 を保持するため，コンテンツタイトル数が多くなるとアクセス数が多いコンテンツのみがキャッシュされることになる．SNS や CGM の特性からアクセス数が多いコンテンツを見た視聴者は，次は関連する動画を視聴する機会が多いが，この要求は前提 3 では満たせない．

TCR ではキャッシングおよびルーティングのキーをタグにすることで少ないキャッシュで同一タグで有りながら他の視聴者の視聴状況も含めた時間的変化に追従する方法が実現できる．

6.1.2 評価環境の妥当性

4.5 節のツリー状構成は, 現実を考慮した評価モデルである．考慮がされている事項と不十分な事項を記載する．

考慮不十分な事項については，タグランキング分布が Zipf 状となると予想されることからスケールモデルとして扱えると考える．スケールモデルでは，各 TCR が持つキャッシュサイズを拡大することで現実的なアクセス状況に対応できると考えるため評価環境は妥当と判断する．

現実世界に即した事項

ネットワーク規模は国内の CMS サーバを考慮した経路数である．ルータ数 $nRouters$ は，この経路数で BA ネットワークを構成する数である．現実の視聴環境は，CMS から視聴端末側に行くほど分岐するため BA トポロジーは妥当と考える．

コンテンツに割り当てられるタグ数 $ntag$=8 はニコニコ動画 [14] の動画ランキング上位 50 位のタグ数を参考にした．

タグアクセス分布の割引率 d=6%は生起確率が各順位で同等になるので，Zipf 分布のバリエーションを網羅している．

現実世界の反映が不十分な事項

タグ総数 N，上位 K 位 K，コンテンツ数 $nContents$ は現実に比べ少ない値であるが，それぞれを現実的にした場合にもタグアクセスランキング分布は Zipf 状と想定されるため，TCR ルータのキャッシュ容量をスケールアップすれば対応できると考える．

CMS や SNS からの推奨コンテンツのリコメンデーションによる効果の確認および視聴端末が推奨コンテンツをネットワークへ推奨する Push 機能の効果については，CMS または SNS 運用方針や視聴者の振舞いのモデルが多様となるため，シミュレーションでの確認は未実施である．

時間的コンテンツアクセス分布は，現実には 2.6.4 節で述べたブーム型，ロングテール型，またはコンスタント型が混在するが，本評価ではコンテンツの時間的アクセス分布までは考慮していない．これらの混在した状況に適した名前解決とキャッシングのポリシについては課題事項であるが，本評価では 1 時間当たりのダウンロード要求を想定しているため，この範囲での評価は各タグランキングの生起数で評価可能と判断する．実サイトのコンテンツランキング集計では毎時，デイリー，週間，月間，合計および新着などの周期的集計があり [109]，数時間以上のシミュレーションを想定する場合は，タグアクセス分布の時間的変化の考慮が必要である．

6.1.3 キャッシュ登録および廃棄方式

本評価では，階層キャッシュ登録方式 LCD, LCE, MCD や LRU, FIFO の影響を見たが，タグアクセスランキングの生起確率はランキング順位が下がると急激に低下することから，random 方式での階層キャッシュ登録・廃棄は不利と考え，本評価では random キャッシュ登録および廃棄方式を用いなかった．LRU 方式では，通信経路上のキャッシュに同一エントリが重複する可能性が高くなるため，端末側は LRU で，CMS 側は別の方式の組み合わせがストレージ経済性に寄与する可

能性がある．今後，階層位置によるキャッシュ登録・廃棄方式の組み合わせによる評価を検討したい．

6.2 研究の有用性

通信経路上のルータにキャッシュを設けてタグによりキャッシュ・経路選択する方法により，動的に変動する CGM 動画コンテンツのタグアクセス分布に適した配信基盤を提案しシミュレーションで効果を確認した．

タグアクセス分布，コンテンツに割り当てるタグ数，ルータのキャッシュ容量による比較を行ない効果を発揮する値を明らかにした．

従来のキャッシュを用いない方式，URI によるキャッシュ方式とタグによるキャッシュ方式を総合評価し優位性の程度を確認した．

本評価のネットワーク規模は，ルータ数を 100 とし，日本国内にコンテンツサーバがあることを想定した．TCR アーキテクチャは，視聴端末の近くにコンテンツ情報およびデータをキャッシュするためネットワーク規模が拡大してもダウンロード速度の改善効果は期待できる．

タグランキング分布は，タグ数，視聴者数によらず Zipf 状となることが予想されるため，TCR では，TCR のキャッシュサイズを調整することでアクセスによるスケーラビリティが保てると考える．

FIA（Future Internet Architecture）の研究事例では，名前解決に特化した NDN やコンテンツを中心に検討した CCN があるが，TCR アーキテクチャは名前解決とコンテンツキャッシュを経路上の TCR ルータで実施し，CCN アーキテクチャにおいてタグ検索が多用される CGM 動画視聴のネットワーク負荷軽減とコンテンツ探索に効果を発揮するアーキテクチャである．この技術は，NDN および他の CCN アーキテクチャと混在または統合可能であり，今後，利用が拡大するソーシャルネットワークを中心としたコンテンツ配信基盤の技術となりうる．

6.3 将来の展望

インターネット上のコンテンツの取得を言語，メディア・サービスの種類，アクセス状況により，検索性能（適合率と再現率）を向上させる取り組みが Semantic Web や FIA で研究されている，TCR のようにネットワーク経路上にキャッシュするアーキテクチャでもキャッシングポリシ，ルーティングポリシの検討により検索性能の向上が期待できる．

TCR の実装は，既存の OSI 参照モデルトランスポート層以下を中心としたネットワークルータにキャッシュを増設しアプリケーション層の処理を追加することにより実現できる．このため現状利用されているインターネットに小規模のルータ数から導入していくことが可能である．

また，TCR アーキテクチャは Linux サーバや視聴端末への導入もルータへの導入と同様に実現できるため，コンテンツ供給側およびコンテンツ視聴側の振舞いを実装することで新たな CCN のニーズに対応できる可能性がある．

提案したタグキャッシングアーキテクチャ TCR は，CGM コンテンツ視聴に特化した FIA，CCN の一部であり，名前解決や経路選択は先行研究で提唱されているアーキテクチャと組み合せ可能である．今後は，先行研究事例との接続性も検討したい．

6.4 まとめ

タグをキーとして経路上に視聴コンテンツ情報，コンテンツデータおよび転送先ルータをキャッシュすることで，URI をキーとした場合に比べて低コストでコンテンツダウンロード時間の改善が図ることができる．特にタグアクセス分布が，順位が増えるにつれて急激に低下する場合に速度改善の効果が大きい．

評価した環境では，キャッシュしない従来のエンドツーエンド方式より，URI によるキャッシュが効果があり，更にタグによるキャッシュ方式が効果がある．タグで検索する割合を増やすとキャッシュヒット率が改善できる．タグランキング分布やタグ割当数により性能は影響を受けるが，いずれの場合でも URI によるキャッシュ方式にくらべて応答性，安定性，および経済性に優れている．

以上から推奨コンテンツを CMS 側で集中的に集計せず，かつキャッシングルータ間の状態を通信しなくても視聴状況に応じて適切に調整するコンテンツ配信基盤のアーキテクチャをシミュレーション結果より提示した．

TCR アーキテクチャは，FIA，CCN のアーキテクチャを 6.1.1 節の前提を排除することにより CGM 動画視聴に特化したコンテンツ配信基盤に適したアーキテクチャである．TCR アーキテクチャは SNS や CCN の新たなニーズに対して，現状のインターネットに大きな変更を行わずとも対応できる．また，下位レイヤや名前解決機構を研究されている FIA や CCN アーキテクチャ組み合わせることで，FIA や CCN とも混在できる可能性がある．

（意図的な空白ページ）

付録 A シミュレータ資源

A.1 評価方法

シナリオは，100 ルータのネットワークで，300 ユーザが 250MB のファイルを平均経路長=7.2 のサーバよりシミュレーション時間あたり 2,000 から 16,000 ダウンロードする際の実行時間およびメモリ使用量を測定する．

A.2 実行時間とメモリ使用量

図 A.1 にシミュレーションの実行時間とメモリ使用量を示す．

実行環境は，CPU：Intel Core i5（3.3GHz）, 32GB メモリ, Linux64bit 版を使用した．

シミュレータの実行時間および使用メモリは共にキューイング回数に比例した．16,000 ダウンロード/時間の場合キューインク回数は 151.2×10^6 回で，その際の実行時間は，1730 秒，使用メモリ 21.9Gbyte であった．使用メモリが大きいのは，転送フラグメント単位で SimPy のインスタンスを作成していることによる．

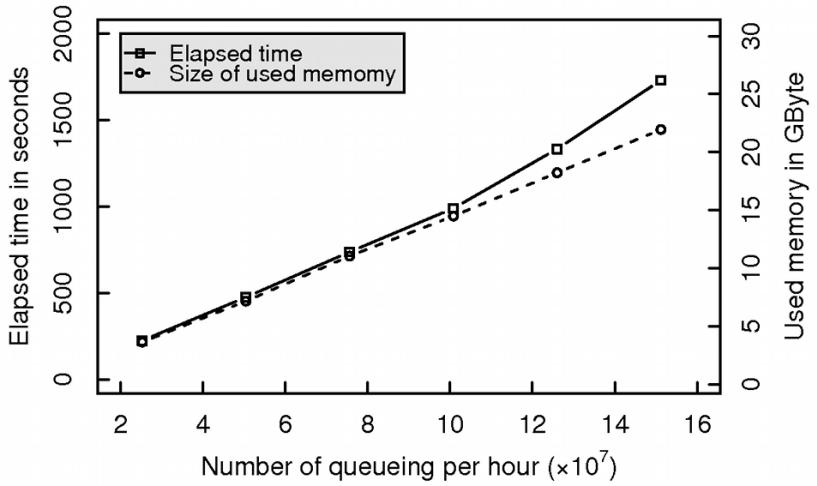


図 A.1 実行時間とメモリ使用量

（意図的な空白ページ）

謝辞

本論文をまとめるにあたり熱心にご指導いただきました国立情報学研究所情報学プリンシプル研究系 山田茂樹教授に心より感謝いたします．

ニュージャージー州工科大学計算機科学科 Cristian Borcea 准教授には，シミュレーションの条件について指導いただき国際会議提出論文のアドバイスをいただきました．愛知県立大学大学院情報科学研究科 奥田隆史教授には理解を深める質問をいただきました．国立情報学研究所アーキテクチャ科学研究系 計 宇生教授，福田健介准教授，鯉淵道紘准教授には中間発表においてアーキテクチャ方式や評価方法についてご指導いただきました．

早稲田大学大学院国際情報通信研究科 田中良明教授，早稲田大学国際情報通信研究センター 矢守恭子客員准教授，ザニケエフ マラット招聘研究員．芝浦工業大学電子情報システム学科 三好 巧教授および出席された大学生・院生の皆様には 3 回参加させていただきました合同合宿セミナーにて，利用面，経済性，検索技術など多方面からのアドバイスをいただきました．

金沢工業大学大学院工学研究科知的創造システム専攻 服部進実教授，金沢工業大学工学部情報工学科 中沢実教授には修士研究の指導いただき，博士課程進学の能力を育成いただきました．

論文発行に至りご指導いただきました皆様に感謝いたします．

シミュレーショタの作成・結果集計にあたりオープンソースソフトウェアを使用しました．有用なソフトウェアを開発・保守されております世界中の皆様に感謝いたします．

最後に独立行政法人科学技術振興機構職員の皆様には業務面でご協力いただきましたことを感謝いたします．

（意図的な空白ページ）

参考文献

[1] Oreilly Tim. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2007. Communications & Strategies, No. 1, p.17, First Quarter, Available at SSRN: http://ssrn.com/abstract=1008839.

[2] I. Peters. *Folksonomies: Indexing and Retrieval in Web2.0*. K G Saur Verlag Gmbh & Co, 2009. 978-3598251795.

[3] 黒瀬浩, 山田茂樹, Cristian Borcea. TCR:フォークソノミータグに適した CGM コンテンツキャッシュルータアーキテクチャ. 電子情報通信学会論文誌. B, 通信, Vol. 96-B, No. 2, pp. 71–82, February 2013.

[4] H. Kurose. Design and performance evaluation of tag caching router architecture for CGM content. In *Computing, Communications and Applications Conference (ComComAp), 2012*, pp. 54 –60. IEEE, January 2012.

[5] 黒瀬浩, 山田茂樹. キャッシュサーバを用いた CGM コンテンツのトラヒック軽減方法の検討. 電子情報通信学会技術研究報告. NS, ネットワークシステム (コンテンツ配信), Vol. 109, No. 448, pp. 177–182, February 2010.

[6] G. Caldarelli, A. Capocci, P. De Los Rios, and M. A. Muñoz. Scale-Free Networks from Varying Vertex Intrinsic Fitness. *Phys. Rev. Lett.*, Vol. 89, p. 258702, December 2002.

[7] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. FolkRank: A ranking algorithm for folksonomies. In *UNIVERSITY OF HILDESHEIM, INSTITUTE OF COMPUTER SCIENCE*, pp. 111–114, 2006.

[8] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 1, pp. 126 –134 vol.1, March 1999.

[9] Pukiwiki. http://pukiwiki.sourceforge.jp/.

[10] Plone. http://plone.org/.

[11] アットウィキ 無料 wiki レンタルサービス. http://atwiki.jp/.

[12] Wikipedia The Free Encyclopedia. http://www.wikipedia.org/.

[13] YouTube - broadcast yourself. http://www.youtube.com/.

[14] ニコニコ動画. http://www.nicovideo.jp/video_top/.

[15] Last.fm. http://last.fm/.

[16] Flickr - Photo Sharing. http://www.flickr.com.

[17] イラストコミュニケーションサービス pixiv. http://www.pixiv.net/.

[18] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[19] Satoshi Niwa, Takuo Doi, and Shinichi Honiden. Web Page Recommender System based on Folksonomy Mining for ITNG '06 Submissions. In *Proceedings of the Third International Conference on Information Technology: New Generations*, ITNG '06, pp. 388–393, Washington, DC, USA, 2006. IEEE Computer Society.

[20] ソーシャル・ネットワーキング サービス mixi. http://mixi.jp/.

[21] twitter. http://twitter.com/.

[22] D.Wessels and K. Claffy. RFC 2016 Internet Cache Protocol (ICP), Version 2, September 1997.

[23] K. Claffy and D. Wessels. RFC 2197 Application of Internet Cache Protocol (ICP), version 2, 1997.

[24] Kisup Kim, Hyukjoon Lee, and Kwangsue Chung. A distributed proxy server system for wireless mobile Web service. In *Information Networking, 2001. Proceedings. 15th International Conference on*, pp. 749 –754. IEEE, 2001.

[25] Tian-Cheng HU, Yasushi IKEDA, Minoru NAKAZAWA, and Shimmi HATTORI. Total Cost-Aware Proxy Caching with Cooperative Removal Policy. *IEICE Transactions on Communications*, Vol. E86-B, No. 10, pp. 3050–3062, October 2003.

[26] Mitsuru ISHII and Shimmi HATTORI. A Proposal of Effective Cooperative Caching System Based on Random Access Assumption. *IEICE Transaction on Cummunication Letter*, Vol. E87-B, No. 6, June 2004.

[27] Tony Bourke, 上谷一訳, 横山晴庸訳. サーバ負荷分散技術. オライリー・ジャパン, 2001. ISBN4-87311-065-3.

[28] Pascal Lorenz, Wenting Tang, Ludmila Cherkasova, Lance Russell, and Matt Mutka. Modular TCP Handoff Design in STREAMS-Based TCP/IP Implementation. Networking — ICN 2001, Vol.2094, Lecture Notes in Computer Science, pp. 71–81. Springer Berlin / Heidelberg.

[29] Mohit Aron, Peter Druschel, and Willy Zwaenepoel. Efficient Support for P-HTTP in Cluster-Based Web Servers. In *Proceedings of the USENIX Annual Technical Conference*, Monterey, California, USA, June 1999. USENIX.

[30] WordPress - Blog Tool, Publishing Platform, and CMS. http://wordpress.org.

[31] WikiCFP - A WIKI for Calls For Pepers. http://www.wikicfp.com/cfp/.

[32] 検索エンジン Google. http://www.google.co.jp/.

[33] buzztter - Snapshot the tweets! http://buzztter.com.

[34] SPYSEE あのひと検索スパイシー. http://spysee.jp/.

[35] David Millen, Jonathan Feinberg, and Bernard Kerr. Social Bookmarking in the Enterprise. *Queue*, Vol. 3, No. 9, pp. 28–35, November 2005.

[36] Delicious (del.icio.us until Aug.2008). `http://www.delicious.com`.

[37] はてなブックマーク. `http://b.hatena.ne.jp/`.

[38] citeulike. `http://www.citeulike.org/`.

[39] Martin Szomszor, Ciro Cattuto, Harith Alani, Kieron O'Hara, Andrea Baldassarri, Vittorio Loreto, and Vito D.P. Servedio. Folksonomies, the Semantic Web, and Movie Recommendation. In *4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0*, June 2007. Event Dates: 3-7th, June 2007.

[40] James Sinclair and Michael Cardew-Hall. The folksonomy tag cloud: when is it useful? *J. Inf. Sci.*, Vol. 34, No. 1, pp. 15–29, February 2008.

[41] Owen Kaser and Daniel Lemire. Tag-Cloud Drawing: Algorithms for Cloud Visualization. *CoRR*, Vol. abs/cs/0703109, , 2007.

[42] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications*, Vol. 4011 of *Lecture Notes in Computer Science*, pp. 411–426. Springer Berlin / Heidelberg, 2006. 10.1007/11762256_31.

[43] Lucia Specia and Enrico Motta. Integrating Folksonomies with the Semantic Web. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *The Semantic Web: Research and Applications*, Vol. 4519 of *Lecture Notes in Computer Science*, pp. 624–639. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-72667-8_44.

[44] Valentina Zanardi and Licia Capra. Social ranking: uncovering relevant content using tag-based recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pp. 51–58, New York, NY, USA, 2008. ACM.

[45] 北研二, 津田和彦, 獅々堀正幹. 情報検索アルゴリズム. 共立出版, 2002. ISBN：978-4－320-12036-5.

[46] Robert Wetzker, Carsten Zimmermann, Christian Bauckhage, and Sahin Albayrak. I tag, you tag: translating tags for advanced user models. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pp. 71–80, New York, NY, USA, 2010. ACM.

[47] Xu Cheng, C. Dale, and Jiangchuan Liu. Statistics and Social Network of YouTube Videos. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229 –238, June 2008.

[48] Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag Recommendations in Folksonomies. In Joost Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors, *Knowledge Discovery in Databases: PKDD 2007*, Vol. 4702 of *Lecture Notes in Computer Science*, pp. 506–514. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-74976-9_52.

[49] ニコニコ大百科. `http://dic.nicovideo.jp`.

[50] Swaminathan Sivasubramanian, Gustavo Alonso, Guillaume Pierre, and Maarten van Steen. GlobeDB: autonomic data replication for web applications. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pp. 33–42, New York, NY, USA, 2005. ACM.

[51] Kei OHNISHI, Kaori YOSHIDA, and Yuji OIE. Folksonomical P2P File Sharing Networks Using Vectorized KANSEI Information as Search Tags. *Transactions of Information and Systems*, Vol. E92-D, No. 12, pp. 2402–2415, December 2009.

[52] BitTorrent. `http://www.bittorrent.com/`.

[53] T. Hopfeld, K. Tutschku, F.-U. Andersen, H. de Meer, and J.O. Oberender. Simulative performance evaluation of a mobile peer-to-peer file-sharing system. In *Next Generation Internet Networks, 2005*, pp. 281 – 287, April 2005.

[54] Akamai Technologies. `http://www.akamai.com/`.

[55] K.L Johnson, J.F Carr, M.S Day, and M.F Kaashoek. The measured performance of content distribution networks. *Computer Communications*, Vol. 24, No. 2, pp. 202 – 206, 2001.

[56] Kideok Cho, Munyoung Lee, Kunwoo Park, T.T. Kwon, Yanghee Choi, and Sangheon Pack. WAVE: Popularity-based and collaborative in-network caching for content-oriented networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 316 –321, March 2012.

[57] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pp. 169–182, New York, NY, USA, 2001. ACM.

[58] Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco, Volkmar Lotz, Elena Simperl, Burkhard Stiller, and Theodore Zahariadis, editors. *Towards the Future Internet - Emerging Trends from European Research*. IOS Press, 2010.

[59] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, Jim Thornton, Diana K. Smetters, eichuan Zhang, Gene Tsudik, kc claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. Named Data Networking (NDN) Project. `http://www.parc.com/content/attachments/named-data-networking.pdf`, October 2010.

[60] Fall Kevin. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pp. 27–34, New York, NY, USA, 2003. ACM.

[61] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. In *Innovations in NGN: Future Network and Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference*, pp. 367 –374, May 2008.

[62] E.J. Rosensweig and J. Kurose. Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks. In *INFOCOM 2009, IEEE*, pp. 2631 –2635. IEEE, April 2009.

[63] Mikko Juhani Pitkänen and Jörg Ott. Redundancy and distributed caching in mobile DTNs. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, MobiArch '07, pp. 8:1–8:7, New York, NY, USA, 2007. ACM.

[64] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant MANETs. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, pp. 32–40, New York, NY, USA, 2007. ACM.

[65] Mirco Musolesi and Cecilia Mascolo. A framework for multi-region delay tolerant networking. In *Proceedings of the 2008 ACM workshop on Wireless networks and systems for developing regions*, WiNS-DR '08, pp. 37–42, New York, NY, USA, 2008. ACM.

[66] Matthew Seligman, Kevin Fall, and Padma Mundur. Alternative custodians for congestion control in delay tolerant networks. In *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, CHANTS '06, pp. 229–236, New York, NY, USA, 2006. ACM.

[67] Jacobson Van, Smetters Diana K., Briggs Nicholas H., Plass Michael F., Stewart Paul, Thornton James D., and Braynard Rebecca L. VoCCN: voice-over content-centric networks. In *Proceedings of the 2009 workshop on Re-architecting the internet*, ReArch '09, pp. 1–6, New York, NY, USA, 2009. ACM.

[68] H. Liu, Y. Zhang, and D. Raychaudhuri. Performance Evaluation of the "Cache-and-Forward (CNF)" Network for Mobile Content Delivery Services. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pp. 1–5. IEEE, June 2009.

[69] Jiachen Chen, M. Arumaithurai, Lei Jiao, Xiaoming Fu, and K.K. Ramakrishnan. COPSS: An Efficient Content Oriented Publish/Subscribe System. In *Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on*, pp. 99 –110, October 2011.

[70] Michael Meisel, Vasileios Pappas, and Lixia Zhang. Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, MobiArch '10, pp. 3–8, New York, NY, USA, 2010. ACM.

[71] Xiaohui Gu, K. Nahrstedt, R.N. Chang, and C. Ward. QoS-assured service composition in managed service overlay networks. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pp. 194 – 201. IEEE, May 2003.

[72] Theodore Zahariadis, Petros Daras, Jan Bouwen, Norbert Niebert, David Griffin, Federico Alvarez, and Gonzalo Camarillo. *Towards a Content-Centric Internet*. IOS Press, 2010.

[73] Kensuke Hashimoto, Yumi Takaki, Chikara Ohta, and Hisashi Tamaki. In-network Hop-aware Query Induction Scheme for Implicit Coordinated Content Caching. In *AFIN 2011, The Third International Conference on Advances in Future Internet*, pp. 69–73, August 2011.

[74] Somaya Arianfar, Pekka Nikander, and Jörg Ott. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, pp. 5:1–5:6, New York, NY, USA, 2010. ACM.

[75] R. Jimenez-Peris, M. Patino-Martinez, G. Alonso, and B. Kernme. How to select a replication protocol according to scalability, availability and communication overhead. In *Reliable Distributed Systems, 2001. Proceedings. 20th IEEE Symposium on*, pp. 24 –33. IEEE, 2001.

[76] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. *SIGOPS Oper. Syst. Rev.*, Vol. 37, No. 5, pp. 29–43, October 2003.

[77] Yang Li, Tao Lin, Hui Tang, and PengSun. A Chunk Caching Location and Searching Scheme in Content Centric Networking. In *IEEE ICC'12 Next Generation Network Symposium (June 2012)*, Jun. 2012.

[78] N. Laoutaris, S. Syntila, and I. Stavrakakis. Meta algorithms for hierarchical Web caches. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pp. 445–452, 2004.

[79] Wenzhong Li, Edward Chan, Yilin Wang, Daoxu Chen, and Sanglu Lu. Cache Placement Optimization in Hierarchical Networks: Analysis and Performance Evaluation. In Amitabha Das, Hung Pung, Francis Lee, and Lawrence Wong, editors, *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, Vol. 4982 of *Lecture Notes in Computer Science*, pp. 385–396. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-79549-0_34.

[80] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ICN '12, pp. 55–60, New York, NY, USA, 2012. ACM.

[81] Wenzhong Li, Edward Chan, Guofu Feng, Daoxu Chen, and Sanglu Lu. Analysis and performance study for coordinated hierarchical cache placement strategies. *Computer Communications*, Vol. 33, No. 15, pp. 1834 – 1842, 2010.

[82] Giuseppe Rossini and Dario Rossi. Large scale simulation of CCN networks. *14èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, La Grande Motte : France (2012)*, 2012.

[83] Bo Sheng, Qun Li, and Weizhen Mao. Data storage placement in sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pp. 344–355, New York, NY, USA, 2006. ACM.

[84] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *SIGMETRICS Perform. Eval. Rev.*, Vol. 24, No. 1, pp. 160–169, May 1996.

[85] 田坂修二. 情報ネットワークの基礎. サイエンス社, 2003. ISBN4 − 901683-11-X.

[86] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, Vol. 74, pp. 47–97, January 2002.

[87] Apache HTTP Server. `http://www.apache.org/`.

[88] Python Programming Language. `http://www.python.org/`.

[89] Matthew A. Russell, 長尾高弘訳. 入門ソーシャルデータ. オライリー・ジャパン, 2011. ISBN978-4-87311-513-9.

[90] Toby Segaran, 當山仁健訳, 鴨澤眞夫訳. 集合知プログラミング. オライリー・ジャパン, 2008. ISBN978-4-87311-364-7.

[91] K. Wehrle, M. Güneş, and J. Gross. *Modeling and Tools for Network Simulation*. Springer, 2010.

[92] The Network Simulator - NS2. `http://www.nsnam.org/`.

[93] Network Simulatior - NS3. `http://www.nsnam.org/`.

[94] OMNeT++ Network Simulation Framework. http://nsnam.isi.edu/nsnam/index.php/.

[95] The Opportunistic Network Enviromnent Simulator. http://www.netlab.tkk.fi/tutkimus/dtn/theone/.

[96] A scalable chunk-level simulator of Content Centric Network(CCN). http://perso.telecom-paristech.fr/~drossi/index.php?n=Software.ccnSim.

[97] memcached -a distributed memory object caching system. http://memcached.org/.

[98] Kenneth Calvert, Matthew B. Doar, Ascom Nexion, Ellen W. Zegura, Georgia Tech, and Georgia Tech. Modeling Internet Topology. *IEEE Communications Magazine*, Vol. 35, pp. 160–163, 1997.

[99] The igraph library. http://igraph.sourceforge.net/.

[100] SciPy. http://www.scipy.org/.

[101] Simulation in Python. http://simpy.sourceforge.net/.

[102] The R Project for Statistical Computing. http://www.r-project.org/.

[103] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The end-to-end effects of Internet path selection. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pp. 289–299, New York, NY, USA, 1999. ACM.

[104] Aiguo Fei, Guangyu Pei, Roy Liu, and Lixia Zhang. Measurements On Delay And Hop-Count Of The Internet. In *in IEEE GLOBECOM'98 - Internet Mini-Conference*, 1998.

[105] D. Rossi and G. Rossini. Caching performance of content centric networks under multi-path routing (and more). In *Technical report*. Telecom ParisTech, 2011.

[106] SQLAlchemy - The Python SQL Toolkit and Object Relational Mapper. http://www.sqlalchemy.org/.

[107] SQLite - Small.l Fast. Reliable. Choose any three. http://www.sqlite.org/.

[108] Daniel A. Menascé, Virgilio A. F. Almeida, and Larry W. Dowdy. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, 2004.

[109] ニコニコ動画内のランキング. http://dic.nicovideo.jp/a/ランキング.

（意図的な空白ページ）

研究業績

博士論文の審査対象

ジャーナル論文（査読あり）

黒瀬 浩，山田 茂樹，クリスチャン ボルセア, ”TCR: フォークソノミータグに適した CGM コンテンツキャッシュルータアーキテクチャ, ” 電子情報通信学会論文集和文 B, Vol.J96-B, No.2, pp.71–82, 電子情報通信学会，2013 年 2 月.

国際会議（査読あり）

Kurose, H., ”Design and performance evaluation of tag caching router architecture for CGM content, ” *In Proceedings. Computing, Communications and Applications Conference (ComComAp) 2012*, pp.54–60, IEEE, Hong Kong, China, Jan. 2012.

一般発表

黒瀬 浩, 山田 茂樹, ”SimPy を用いた待ち行列シミュレーション, ” B-6-63, 電子情報通信学会通信ソサイエティ大会, 2012 年 9 月.

黒瀬 浩, 山田 茂樹, ”キャッシュサーバを用いた CGM コンテンツのトラヒック軽減方法の検討, ” 信学技報（NS, ネットワークシステム）, Vol.109, No.448, pp.177–182, 電子情報通信学会, 2010 年 2 月.

（ポスター発表）黒瀬 浩, 山田 茂樹, ”フォークソノミーを用いたキャッシュルーターの検討と評価, ” 国立情報学研究所平成 24 年度オープンハウス, 2012 年 6 月.

（ポスター発表）黒瀬 浩, ”キャッシュルータを用いた CGM コンテンツ流通基盤, ” 第 4 回総研大ワークショップ, SWS2010-B01-03, 2010 年 10 月.

（ポスター発表）黒瀬 浩, 山田 茂樹, ”A Self-organized Network using Content-driven Caching Routers for CGM Content, ” 国立情報学研究所平成 22 年度オープンハウス, 2010 年 6 月.

参考

博士課程入学以前の業績

黒瀬 浩, 服部 進実, “フォークソノミーと Wiki を用いた課題解決プロセス活性化支援システム, ” 情処技報 (グループウェアとネットワークサービス), Vol.2008, No.91, pp.77-82, 情報処理学会, 2008 年 9 月.

荒井 滋人, 黒瀬 浩, 野村 宣生, 服部 進実, “ CGM コンテンツの可視化を目指した権利継承システム, ” 情処技報（グループウェアとネットワークサービス）, Vol.2009, No.33, pp.121-126, 情報処理学会, 2009 年 3 月.

松嶋 和気子, 黒瀬 浩, 中沢 実, 服部 進実, “集合知を用いた自己管理学習システムの検討, ” 情処技報（マルチメディア通信と分散処理）, Vol.2007, No.117, pp.1-6, 情報処理学会, 2007 年 11 月.

（査読あり）黒瀬 浩, 中沢 実, 服部 進実, “災害対策としてのＳＱＬデータベース複製方法の実装および性能評価”, DICOMO2006 シンポジウム（システム性能評価セッション） 情報処理学会シンポジウムシリーズ, Vol.2006, No.6, pp.761-764, 情報処理学会, 2006 年 7 月.

黒瀬 浩, 中沢 実, 服部 進実, “自律的多段ＤＢ複製方法の実装および評価”, 信学技報（ディペンダブルコンピューティング）, Vol.105 No.607, pp.31-36, 電子情報通信学会, 2006 年 2 月.

TCR: CGM コンテンツに適したタグキャッシングルータアーキテクチャの研究
著 者　黒瀬 浩
発 行　2013 年 3 月