MobiCCN: Mobility Support with Greedy Routing in Content-Centric Networks

Liang Wang, Otto Waltari, Jussi Kangasharju

Department of Computer Science, University of Helsinki, Finland

***Abstract*—Content-Centric Network (CCN) shifts the Internet from point-to-point paradigm to receiver-driven data-centric paradigm. While it solves many problems in the current Internet and opens the door to many novel applications, it also leaves many challenges unanswered, e.g., mobility support and mobile content publishing and dissemination. In this paper, we show how a greedy routing can be implemented in CCN architecture to support mobility. This allows for efficient content publisher mobility and supports seamless handoffs for interactive connections. We present our solution – MobiCCN, and evaluate it thoroughly in realistic network topologies to show it outperforms other popular mobility schemes.**

I. Introduction

Content-Centric Networking (CCN) [1] shifts the current Internet paradigm from point-to-point communication to content-centric dissemination. CCN solves some problems in current Internet like network congestion, content delivery and security and etc. The essence of CCN is accessing content by name. All content is identified, addressed and retrieved by name instead of physical locations. A client requests a content item by sending out an `Interest` packet into the network. The network nodes can potentially store the content to serve the future requests.

Naming also poses significant challenges on routing. In current Internet, DNS handles the mapping between hostnames and IP addresses. However, as far as content is concerned, the number of items to handle is orders of magnitude larger than names handled by DNS. Hierarchical names have been proposed because they offer performance gains through aggregation. However, this assumes the names are correlated with the underlying network topology, and may leave lots of gaps if the content has high mobility. While Caesar et al. showed flat name is feasible in [2], Ghodsi et al. argued in [3] that both hierarchical and flat names are essentially the same.

Due to its receiver-driven design, CCN inherently supports receiver mobility. A lost packet can be recovered by retransmitting the `Interest`. However, sender or publisher mobility is difficult to achieve. One scenario is maintaining the communication between mobile caller and callee in a VoIP call. Another is mobile content publishing and dissemination. In both cases, data sources are highly mobile. The standard CCN requires necessary name operations to handle data source mobility, but updating and propagating names are expensive operations in network. Performance and scalability are hard to achieve in such cases.

All these issues boil down to the question how we route the packets to the correct destination. Routing and name resolution form the core of the problem. An interesting direction in routing research is greedy routing which was first applied to mobile ad-hoc network, but it can also be used in traditional wired network. Zhang et al. in [4] presented a greedy routing based on underlying metric space. The idea is that each router is assigned a coordinate from a name-based metric space. The router makes the forwarding decisions based on the distance of its directly connected neighbors and the destination in the packet. CAIDA has also done experiments in evaluating greedy routing, but they have been limited to small-scale and manually assigned (geographical) coordinates [5].

In this paper, we show that by introducing a greedy routing into CCN architecture, we can provide an efficient mechanism for seamless mobility, also solve the disparity between enormous space of names and scarce routers' resources. Even though we focus the discussion on CCN, it is worth pointing out that our solution can be equally applied to other similar systems. Our contributions are as follows:

1) We show that greedy routing can be implemented as routing policy in CCN.
2) We present MobiCCN, our mobility scheme, and evaluate it thoroughly in realistic settings.
3) We compare MobiCCN with other schemes from literature, and show that it outperforms them.

The paper is organized as follows. Section II gives a brief introduction on CCN and greedy routing techniques. We present MobiCCN in Section III and evaluate it in Section VI. Section V gives a comparison of common mobility schemes in CCN. Finally, Section VI concludes the paper.

II. Background

A. CCN

Information-Centric Networking (ICN) has been an active research area for several years. There are several similar independent proposals [1], [6], [7], and all of them are essentially based on the similar concept – accessing content by name. We use CCN [1] as basis for our work, but many of the key elements of our solution can be applied in other ICN architectures as well.

CCN is a receiver-driven model. To retrieve a content, the user needs to construct an `Interest` packet first, which contains the content name. The `Interest` is sent to the

network and routed in a hop-by-hop manner by CCN routers. Each router checks the content name and if there is a copy in the local storage – *Content Store* (`CS`), the response will be sent back immediately. Otherwise, the router checks its *Forwarding Information Base* (`FIB`) and uses longest prefix matching to determine which `face` the `Interest` will be forwarded to. The forwarded `Interest` leaves an entry in router's *Pending Interest Table* (`PIT`). If the `Interest` finally reaches the data source and is replied by the server, the response can follow the entries in `PIT` left by the previous `Interest` and goes back to the user.

CCN inherently supports receiver-side mobility. To recover a lost packet during mobility, receiver only needs to retransmit the previous `Interest`. Intermediate routers can use the copies in their `CS` to serve the request. However, if the data source is mobile, retrieving data will be much more difficult. Common schemes use hierarchical naming and if a data source moves into a new domain, it has to perform expensive name operations to handle the mobility. In some schemes, the receiver needs to be informed about the changes so that communication can continue. In [8], Kim et al. compared several mobility schemes, including their own solution.

B. Greedy Routing

Greedy routing has a long history in mobile and sensor networks. In such networks, a node does not have global knowledge of the network topology and only knows its neighbors. Greedy routing makes it possible to route packets in the "dark" by assigning coordinates to nodes. However, greedy routing doesn't specify its underlying metric space, node usually uses its actual geographical coordinates as its locator, which is also referred as geographical routing. The destination's coordinate is embedded into the packet header. To forward a packet, a node calculates the distance between the destination and each of its directly connected neighbors, and selects the one closest to the destination as the next hop. However, using geographical coordinates cannot guarantee 100% delivery due to the *local minimum* issue. In a connected graph, *local minimum* refers to the situation that a node x itself is closer to the destination y than any of its neighbors even though y is not x's directly connected neighbor. In this case, the node cannot decide who should be the next hop therefore routing fails. In [9], Cvetkovski et al. proposed Gravity-Pressure routing to provide optional path when local minimum occurs.

Using geographical coordinates is equivalent to embedding the network into Euclidean space. However, Euclidean space is not the only candidate for greedy routing's underlying metric space. Instead of real geographical coordinates in Euclidean space, nodes can use virtual coordinates from any well-defined metric space. Therefore, another solution to the local minimum problem is to embed the topology in a "better" metric space. The idea is to find a *greedy embedding* for arbitrary topologies. Greedy embedding refers to the embedding with the property that given any destination y which is not directly connected to a node x, x can always find a neighbor of him who is closer to y than himself. Kleinberg gave a proof in [10], showing that if we use a hyperbolic space as the underlying metric space, then every connected graph has a greedy embedding. Therefore 100% delivery is guaranteed. In [9], Cvetkovski et al. extended Kleinberg's work to dynamic graphs.

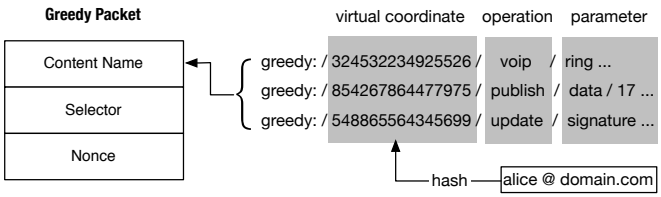


Fig. 1: Greedy packet type. All the greedy packets are normal CCN `Interests`, MobiCCN only reserves the prefix *greedy:/* to activate the greedy protocol.

III. Architecture

In this section, we present our MobiCCN by describing how communication happens in the system. Then we give a specific scenario of VoIP in Section III-C.

A. Proposal

As we have seen, the conventional way of routing and name resolution in CCN is incapable of handling sender mobility issues. We propose a new routing protocol which can coexist with the standard CCN routing protocol.

There are two routing protocols in MobiCCN, the standard CCN protocol and the greedy protocol. MobiCCN neither changes the existing packet format nor adds any new ones. A greedy packet is just a normal CCN `Interest`. We only reserved prefix *greedy:/* for greedy protocol, while the standard one uses *ccnx:/*. Whenever a router receives a packet with the name starting with *greedy:/*, it switches to greedy protocol to forward the packet. As Figure 1 shows, the general format of the name of a greedy packet is *greedy:/vc/operation/parameters/...*.

Each router is assigned a *vc (virtual coordinate)* from the underlying hyperbolic space $\mathbb{H}$ (using MWST algorithm discussed in Section IV-D). Coordinate allocation can be done in many ways, e.g., manually or by the ISP using the MWST algorithm (or any similar embedding algorithm). As shown in [9], coordinate allocation can also be done automatically and dynamically whenever a new node joins or an old node leaves by using the dynamic embedding algorithm in [9].

Greedy protocol uses *vc* as the destination address and embeds it into the content name of a packet. Each router only maintains a small table of its neighbors' coordinates. In order to forward a packet, the router first extracts destination coordinate *vc* from the packet, then it calculates the distance between the destination and each of its neighbors. The packet is forwarded to the neighbor who is closest to the destination.

However, we do not have to calculate distance for every greedy packet and can reduce the computational overheads by caching previous results. When a greedy packet arrives,

Algorithm 1 Greedy routing protocol - Interest

Input: Greedy Interest GI
Output: Forward decision
$dst \leftarrow$ destination coordinate of GI
if $(dst, face)$ in FIB **then**
  $oface \leftarrow face$
else
  for each directly connected neighbor N_i **do**
    $d_i \leftarrow$ distance between dst and N_i
  end for
  $oface \leftarrow N_i$ with smallest d_i
  Update $(dst, oface)$ pair in FIB
end if
Forward GI to $oface$

Algorithm 2 Greedy routing protocol - Update

Input: Greedy Update GU
Output: Update FIB of the routers between a mobile user and its corresponding rendezvous point
$iface \leftarrow$ ingress of GU
$dst \leftarrow$ destination coordinate of GU
Update $(dst, iface)$ pair in FIB
for each directly connected neighbor N_i **do**
  $d_i \leftarrow$ distance between dst and N_i
end for
$oface \leftarrow N_i$ with smallest d_i
$d \leftarrow$ destination between dst and current router
if $d < min(d_i)$ **then**
  Rendezvous point, stop forwarding
else
  Forward GU to $oface$
end if

the router checks whether there is an entry in `FIB` using the longest prefix matching. If the result is positive, it means the distance has been calculated before, then packet is forwarded to the next hop stored in `FIB`. The router's performance is independent of the number of greedy packets passing by.

Each user has a unique ID to identify himself, which can be the same as his CCN name. Greedy routing then maps the user ID into the same hyperbolic space $\mathbb{H}$ to obtain its virtual coordinate. Mapping can be done with any well-defined hash function, e.g. SHA-1. Each user has a dedicated router who is closest to him in $\mathbb{H}$ as his *host router* in the network. The host router serves as rendezvous point and relays traffic for him. [1]

Whenever a data source moves to a new attachment point, it sends out an `Update` packet to its host router. The `Update` has a name like *greedy:/vc/update/...*, indicating it is an update operation. Each router the `Update` passes by will update the corresponding entries of that data source in its `FIB` accordingly; then `Interests` towards the source can be forwarded correctly to the new domain. From receiver's perspective, it always uses data source's original name to communicate. So there is no need to change content name even after the source's handoff, because the `Interests` can always reach source's host router (in the worst case).

Algorithms 1 and 2 show how greedy `Interest` and greedy `Update` are processed in a router. It is not necessary for a greedy `Interest` to arrive at the rendezvous point in order to reach the mobile user, because the greedy `Interest` may pass a router which already cached $(dst, face)$ in `FIB` before reaching the rendezvous point. This means that the stretch in MobiCCN can even be lower than that in normal greedy routing, however this advantage is highly topology-dependent and not a guaranteed property.

B. Security

As shown in [1], CCN is built on the notion of *content-based security*. Each piece of content can be authenticated by the digital signature embedded in the packet header. MobiCCN inherently embodies the CCN's security, and has the equal strength of CCN in against many kinds of attacks.

To prevent malicious users from exploiting `Update` packets to disturb the normal communication, the sender is required to sign every `Update` packet. Whenever an `Update` packet arrives, the router needs to check whether the sender is the actual owner of the name so that he has the right to update his corresponding entries in `FIB`. This can be easily done by validating the digital signature in the packet, if the key used for signing is not the same as the one of the actual owner, the packet should be dropped immediately. Technically, this means signed `Interest` instead of signed Content Object. According to CCNx specification, the signature can either be appended to the content name (as MobiCCN does), or stored in the additional field in the header.

Validation requires extra operations and network communication. To reduce the overheads and also prevent work-factor attack, we adopted three mechanisms in the design. First, router is not obligated to validate every `Update` packet as long as sender's attachment point remains unchanged. Second, only the edge router (sender's attachment point) is responsible for validating sender's authenticity, the downstream routers are free from these operations. Third, owner's key is cached before its expiration so that a router needs not retrieve it all the time whenever it needs validation.

Instead of mandating a *one-size-fits-all* approach, CCN adopts a very flexible *contextual* trust model. Clients should determine themselves whether the data is trustworthy. In a similar vein, MobiCCN design allows both traditional hierarchical **PKI** (public-key infrastructure) and non-hierarchical model like **SDSI** [11]. However, discussion over the trust models is beyond the scope of this paper and is left as our future work.

C. VoIP Scenario

We present a VoIP scenario as an example of how MobiCCN handles mobility. Note that MobiCCN is not limited to voice calls, but is a solution for general publisher mobility in CCN.

[1] There are obvious similarities between MobiCCN and Mobile IP, and the host router is roughly equivalent to a home agent.

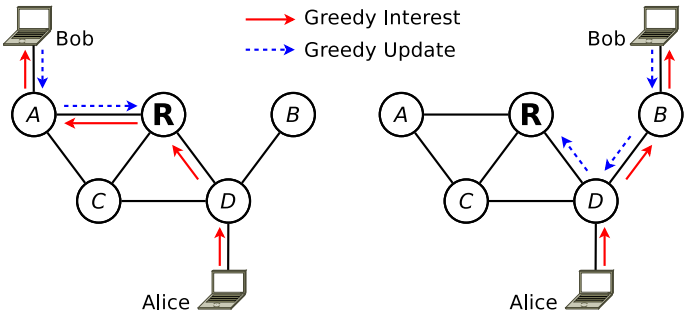


Fig. 2: MobiCCN Scenario. Bob's host router is R and between the left and right figure Bob changes his attachment point from A to B. Router D caches Bob's `Update` and Alice's `Interest` packet on the right-hand figure does not need to go to R but D is able to forward it directly to B.

Figure 2 shows a simple VoIP scenario in MobiCCN with both two users – Alice (Caller) and Bob (Callee). Both Alice and Bob use their email addresses as their unique IDs, and their corresponding virtual coordinates in $\mathbb{H}$ are vc_{alice} and vc_{bob} respectively. Now Alice wants to initialize a VoIP call to Bob. To set up the connection, she first tries the standard CCN way by using the name like *ccnx:/domain/voip/bob*. However, if Bob already starts moving before Alice's attempts, the connection setup will fail. After the timeout event is triggered, Alice activates greedy protocol and sends out a new `Interest` with the name *greedy:/*vc_{bob}*/voip/ring*. The `Interest` will pass by Bob's host router and finally find its way to Bob using greedy protocol. If both fail, then Bob is considered offline.

The previous two-attempt setup only happens once in the beginning of the communication, and the overhead can be avoided if greedy protocol is used as the primary protocol. The standard one is then used as backup protocol to achieve lower average latency if both counterparts are stationary.

Another way to avoid two-attempt setup is for Alice to send out two `Interests` (both standard and greedy one) in parallel in the beginning, then choose the protocol after Bob replies. The overhead is just one extra packet.

When Bob is the data source and he moves into a new domain B, he sends a greedy `Update` with the name *greedy:/*vc_{bob}*/update* after the handoff. The `Update` eventually reaches his host router and all the intermediate routers update their `FIB` in order to forward `Interests` towards Bob correctly.

However, as we can see from Figure 2, Alice's `Interest` reaches D before R, and router D already updated its `FIB` from Bob's greedy `Update`. Alice's `Interest` can then be routed directly towards Bob without passing the host router.

D. Features & Limitations

MobiCCN can either be used as a backup solution for mobility issues, or the primary scheme for mobile content publishing and dissemination. MobiCCN only needs marginal modifications to CCN routers and does not interfere with the standard protocol. Applications using the standard protocol are not aware of the greedy protocol.

Greedy protocol might increase stretch in the communication. However, as we show in Section V, it is still much lower than other popular schemes and stretch can be further reduced by using a better embedding algorithm. Furthermore, due to the flexibility of MobiCCN, users can negotiate with each other to switch to the standard protocol if they stop moving.

Network	Routers	Links	POPs	Diameter	Avg. Path
Exodus	338	800	23	12	5.824
Sprint	547	1600	43	12	5.182
AT&T	733	2300	108	11	6.043
NTT	1018	2300	121	14	6.203

TABLE I: Graph properties of the four selected ISP networks

IV. Evaluation

A. Prototype & Testbed

We implemented MobiCCN prototype in Python. The prototype works similarly to CCN defined in [1]. Greedy routing is implemented as an extension to the standard CCNx routing protocol. We are also implementing MobiCCN in the CCNx prototype as a plug-in.

We chose four real-world ISPs networks to run our experiments: Exodus, Sprint, AT&T and NTT. The network topology files are from Rocketfuel [12] project. For the network with multiple components, we only use the biggest one. Table I shows an overview of the networks with their graph properties.

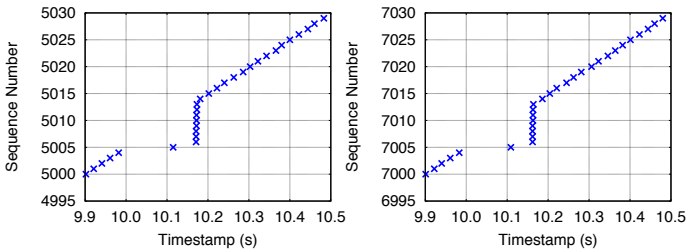
All the experiments are performed on our department cluster consisting of 240 Dell PowerEdge M610 nodes. Each node is equipped with 2 quad-core CPUs, 32GB memory, and connected to a 10-Gbit network. All the nodes run Ubuntu SMP with 2.6.32 kernel. Multiple virtual routers are multiplexed onto one physical node if the ISPs network is larger than the cluster network.

B. Handoff Delay

Handoff delay is one of the most important metrics for evaluating a mobility solution. We experimented our solution on four topologies, but since the results are similar in all of them, we only present the results on AT&T network. We also compared MobiCCN with different mobility schemes. However, since Interest Forwarding has been shown to be superior to the others [8], we only compare against it in this section. Note that the evaluation in [8] is done on a synthetic topology and we now run their algorithm on a real ISP topology.

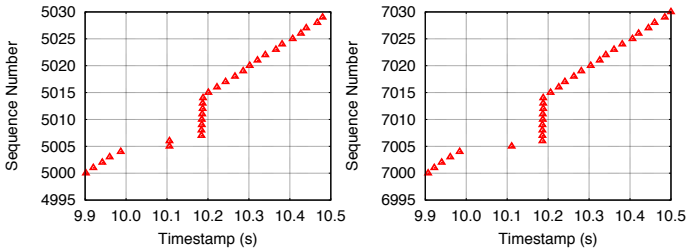
In our simulation, the link delay is set to 5 ms. The initial placement of the sender and receiver is arbitrary. The selection of the next attachment point of the mobile sender is among the nodes within a 2-hop radius. Layer 2 handoff delay is set to 100 ms, and loss detection timer is also set to 100 ms. Both caller and callee perform a simultaneous handoff at 10 sec. Caller and callee send out `Interests` at a rate of 50 pkts/s.

Figure 3a shows the sequence number of the content piece the caller received when simultaneous handoff happened.



(a) Caller (b) Callee

Fig. 3: MobiCCN handoff delay



(a) Caller (b) Callee

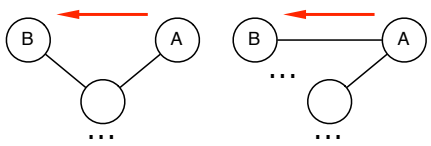
Fig. 4: Interest Forwarding [8] handoff delay

When the caller finished layer 2 handoff at 10.1 sec, he started re-requesting the lost data. Because packet #5005 was already on its way to the caller when the handoff happened, it was already cached by an intermediate router. That is why packet #5005 can be quickly re-transmitted at 10.15 sec just after the layer 2 handoff finished. The rest of the re-transmissions are subject to one RTT, they arrive later at 10.17 sec. The caller's handoff delay is 173 ms.

Figure 3b shows sequence number callee received during the handoff. The callee's handoff delay is 163 ms, which is shorter than the caller's 173 ms. The reason is that paths between caller and callee are not symmetric. Path from callee to caller (6 hops) is shorter than that from caller to callee (7 hops).

Figure 4 shows the handoff delay in Interest forwarding scheme from [8]. The experiments are done with the same setting as that of MobiCCN. The caller's and callee's handoff delays are the same, both are 188 ms. Although small, this difference to MobiCCN is consistently present and measurable in all our experiments. The reason of the longer handoff delay is that the path between caller and callee increased from 6 hops to 8 hops after the handoff. This is shown in Figure 5. If data source moves from *A* to *B*, topology α in Figure 5a will not increase the path length. However, topology β in Figure 5b will increase the path length by 1. Triangular routing cannot be eliminated in Interest Forwarding if user's home agent *A* becomes the next hop in the new path. It is more difficult to prevent this issue if topology β is closer to the network core.

Even though neither MobiCCN nor Interest Forwarding requires users to change names after the handoff, Interest Forwarding may be affected by the network topology.



(a) Topology α (b) Topology β

Fig. 5: Interest Forwarding is subject to topology

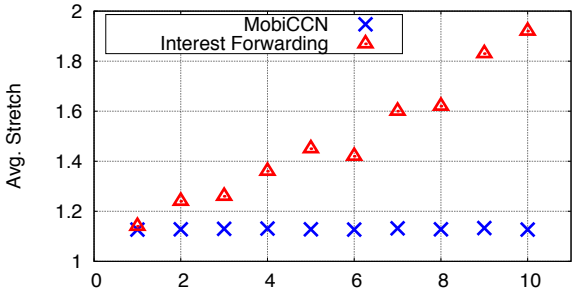


Fig. 6: Average stretch as a function of number of handoffs

C. Scalability

We designed another experiment to see how the network topology affects the path stretch. In the experiment, callee is fixed and caller moves N times. Every t sec, caller moves to a new attachment point. (While such mobility might not happen in many scenarios, it serves to illustrate MobiCCN's scalability even under extremely mobility.) We measured the stretch between caller and callee after each handoff. The experiments were repeated 50 times, and Figure 6 shows the average stretch.

For Interest Forwarding, despite of some small fluctuations, stretch increases while the caller keeps changing its attachment point. The reason is that if the previously attached router is the next hop of the newly attached router, the path will increase after handoff. As the caller moves more, the probability of this happening varies according to the topology, thus causing some fluctuation in the results. However stretch shows a steadily increasing trend. Furthermore, stretch in Interest Forwarding scheme is consistently higher than in MobiCCN, which is stable at 1.13. MobiCCN's performance is independent of moving and topology once the embedding is done.

This experiment implies the network topology can have a significant impact on the performance of a mobility scheme. An artificial topology is incapable of reflecting all the characteristics from realistic topologies, thus evaluations on purely synthetic topologies are likely to yield results that do not correspond to results in a real network topology.

D. Stretch

The host router approach of MobiCCN may increase stretch because traffic in many cases passes through the host router, but a better embedding algorithm can help reduce the stretch.

To embed a network into a hyperbolic space, the first step is to derive a spanning tree from the network, then we embed the

	Exodus	Sprint	AT&T	NTT
Avg. Stretch	1.384	1.375	1.271	1.320
Min. Stretch	1.149	1.197	1.110	1.198
MWST	1.212	1.185	1.128	1.150
Improvement	13.06%	13.82%	11.25%	12.88%

TABLE II: Stretch of four networks with different spanning tree algorithms

tree into the space. Kleinberg showed in [10] that the greedy embedding of a spanning tree of a graph is also the graph's greedy embedding. However we can derive multiple spanning trees from the same graph, and different trees may lead to different stretches. When the network is small, the embedding can be done manually and the stretch can be reduced to as low as 1, like [5]. However, manually assigning coordinates is infeasible for a large network.

In [13], Cvetkovski et al. implemented two heuristics and showed that they can improve the average hop stretch by about 30%. In this paper, we used the Maximum-Weight Spanning Tree (MWST) in [13] to construct the spanning tree on the experiment topology and embedded it into a Poincaré disk.

For each network, we generated 5000 random minimum spanning trees and embedded them into the Poincaré disk. We recorded the average and the minimum value; these are shown in the first and second row in Table II.

We used MWST for the greedy embedding and recorded its stretch and also calculated the improvement MWST achieved compared with the average value. The third and fourth row in Table II show the results. MWST has about 11% – 13% improvement on realistic network topologies.

E. Performance Impact

When CCN router forwards a greedy packet, router spends extra CPU cycles in computing the distances between the destination and its neighbors to decide the next hop. However, since MobiCCN uses the same longest prefix matching mechanism as that in standard CCN, it can utilize `FIB` to cache the previously calculated results to reduce the overheads. Then the CPU overheads become independent of the absolute number of greedy packets passing by, but only a function of the arrival rate of the packets containing new destinations. The `FIB` entries are tentative and will be purged out automatically after predefined expiration time. So even if all the traffic are greedy packets, the overheads still remain at a low level.

We evaluated how greedy routing impacts router performance in the worst case without optimization. In our experiment, we first measured the router's maximum throughput when all the packets passed by are standard CCN packets. Then we increased the fraction of greedy packets step by step, and examined how it degrades CCN router's throughput. Our results show that the throughput drops linearly as a function of the fraction of greedy packets. With up to 10% greedy packets, the drop is negligible, but if the traffic consists of purely greedy packets and the router has to calculate the distance for every packet, the throughput drops by about 30%.

V. Comparison of Mobility Schemes

In this section, we compare MobiCCN with other mobility schemes presented in literature. Kim et al. [8] already performed a basic comparison between Interest Forwarding and the others. In this paper, we evaluate the schemes from the following perspectives: handoff delay, average latency, capability of handling simultaneous handoff, scalability, single point of failure and implementation complexity. Table III summarizes the comparison.

Sender-driven control message is the most straightforward scheme. In this scheme, the moving user sends out a control message explicitly to the receiver to inform his new hierarchical name when handoff occurs. However, this scheme cannot handle well the situation where both sender and receiver are moving. The communication may completely break down when simultaneous handoff happens. Another problem is that receiver must regenerate the new `Interest` for the lost packet using sender's new hierarchical name. The advantages of this scheme are its simplicity and pure CCN style, and average latency in the communication is low. All the modifications are in the application layer.

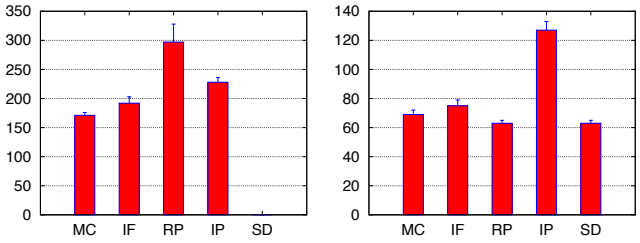
In **Rendezvous point** scheme, user needs to update their attachment point to the rendezvous point periodically or when handoff occurs. If the receiver fails to get response within a predefined time, the receiver will think the data source has changed its attachment point. Then the receiver sends the query to rendezvous point to get the update. In this scheme, the communication will not completely break down if simultaneous handoff happens, but it is still possible that the receiver gets outdated information and suffers from a large delay due to second lookup operations. Furthermore, the receiver still needs to regenerate `Interest` with the new name for the lost packets. Generally, this scheme suffers from a large handoff delay. The advantage is that average latency is low and the modification is on the application layer; lookup only happens when timeout is triggered. Normal communication is done as in CCN.

Indirection point scheme uses separate server to relay all the traffic. If handoff occurs, the `Interests` to the user are buffered first at the indirection point, then forwarded later until the moving user updates the new name to the Indirection point. Because all the traffic must pass the indirection point, the obvious disadvantage is the indirection point becomes the single point of failure and a bottleneck if the traffic load is heavy. Even though the handoff delay can be improved in this scheme, normal traffic suffers from a large average latency.

In all aforementioned schemes, the content must change its name based on the attached domain. However, updating content name is an expensive operation in CCN and this makes seamless handoff difficult to implement. Kim et al. proposed **Interest Forwarding** in [8]. In their scheme, the mobile user must send a notification to the current attached router when it notices a handoff is imminent. The router will start buffering the coming `Interests` for the user. Then the user can fetch the buffered `Interests` by sending a `virtual`

	Avg. Latency	Handoff Delay	Simultaneous Handoff	Scalability	Single Point of Failure	Complexity
MobiCCN	Medium	Low	Yes	High	No	Medium
Sender-Driven Msg	Low	High	No	High	No	Low
Rendezvous Point	Low	Medium	Yes	Low	Yes	Low
Indirection Point	High	Medium	Yes	Low	Yes	High
Interest Forwarding	Medium	Low	Yes	Medium	No	High

TABLE III: Comparison of different mobility schemes. MobiCCN achieves good trade-off point from various perspective.



(a) Handoff delay (ms) (b) Average latency (ms)

Fig. 7: Comparison of latencies in different schemes. MC: MobiCCN, IF: Interest Forwarding, RP: Rendezvous Point, IP: Indirection Point, SD: Sender-Driven Message

`Interest` back to the old attached router. The `virtual Interest` also updates the `FIB` in the intermediate routers so that the following `Interests` can be forwarded correctly. This scheme avoids changing the content name by using tentative home agent. However, one problem is the whole scheme may fail if an imminent handoff becomes hard to predict. Secondly, as we have shown in this paper, the path may grow longer while the user is moving, and the following traffic suffers from the larger latency.

Figure 7 shows the handoff delay and average latency in each scheme. The experiment was repeated 100 times and the average value with standard deviation is presented. In Figure 7a, simultaneous handoff was evaluated. Sender-Driven Scheme is missing because it cannot handle simultaneous handoff. The performance of Rendezvous Point and Indirection Point depends on the placement of the Indirection/Rendezvous server. In our experiment, we deployed the server 6 hops away from both two mobile nodes. MobiCCN has the best performance of all the solutions. Rendezvous Point is the worst and has the largest variation due to the possibility that user may receive outdated information.

Figure 7b shows the average latency in the communication. We let the data source have two handoffs before we start the evaluation. The Rendezvous Point and Sender-Driven Message have the shortest latency because they always use the shortest path. Indirection Point is the worst because all the traffic is relayed. MobiCCN is slightly higher than the best one due to the stretch caused by greedy routing. But the latency can be further reduced and become closer to the one by using better embedding algorithm. Interest Forwarding is a little higher than MobiCCN due to the issue we discussed in Section IV-B.

In summary, MobiCCN outperforms the other solutions in terms of delay and (for the most part) latency.

VI. Conclusion

In this paper, we present how we extend geographical routing in current CCNx to solve mobility and mobile content publishing and dissemination issues in CCN. By embedding network topology into hyperbolic space, we distribute the rendezvous points and name resolution functionality into the network. We compared MobiCCN to other proposed CCN mobility schemes and showed that it outperforms existing schemes both in terms of handoff delay and communication latency. We are currently implementing our solution on CCNx as extension, which is fully compatible with the standard CCN routing protocol.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th ACM Conext*. New York, NY, USA: ACM, 2009, pp. 1–12.

[2] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: Routing on Flat Labels," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 363–374, 2006.

[3] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, 2011.

[4] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project," *Technical Report NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[5] Caida, "Greedy forwarding on the ndn testbed," August 2011. [Online]. Available: http://www.caida.org/research/routing/greedy_forwarding_ndn/

[6] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 181–192, 2007.

[7] Publish/Subscribe Internet Routing Paradigm, "Conceptual architecture of psirp including subcomponent descriptions. Deliverable d2.2, PSIRP project," , August 2008.

[8] D.-h. Kim, J.-h. Kim, Y.-s. Kim, H.-s. Yoon, and I. Yeom, "Mobility support in content centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, 2012.

[9] A. Cvetkovski and M. Crovella, "Hyperbolic embedding and routing for dynamic graphs," in *IEEE INFOCOM*, april 2009, pp. 1647 –1655.

[10] R. Kleinberg, "Geographic routing using hyperbolic space," in *IEEE INFOCOM*, may 2007, pp. 1902 –1909.

[11] R. L. Rivest and B. Lampson, "Sdsi - a simple distributed security infrastructure." MIT, 1996.

[12] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proceedings of ACM SIGCOMM*, 2002.

[13] A. Cvetkovski and M. Crovella, "Low-stretch greedy embedding heuristics," in *Computer Communications Workshops (INFOCOM WKSHPS)*, march 2012, pp. 232 –237.