SOFTSTAGE: Content Staging for Vehicular Content Delivery in the eXpressive Internet Architecture

Jing Wang[†], Chenren Xu[†✉], Xiuming Wang[*], Wangyang Li[†], Zhenyi Li[†], Shuang Jiang[†], Peter Steenkiste[§]

[†]Peking University [*]South China University of Technology [§]Carnegie Mellon University

***Abstract*—Client mobility is a fundamental challenge when accessing the current Internet, especially in the context of vehicular networking because of its intermittent connectivity nature. Meanwhile, today's network applications are evolving from host-to-host communication to content retrieval, and fostering new designs of Information-centric networking (ICN) protocol and system optimized towards this end. In this paper, we present SOFTSTAGE, a client instructed ICN-based network layer function that effectively manages the edge caching to perform reactive content staging to improve vehicular content delivery without any assumption about the client mobility pattern. Experimental results based on an implementation in eXpressive Internet Architecture (XIA) shows that SOFTSTAGE achieves up to 10x throughput gain in vehicular networking environments.**

I. INTRODUCTION

Mobile access has become the norm for today's Internet, however mobile user experience continues to be inferior to that of the wired Internet. Take vehicular to infrastructure (V2I) networking as an example. Intermittent network connectivity, as a result of coverage gaps and frequent (slow) handoffs at layers 2/3, does not only affect performance but also disrupts communication sessions at the transport and application layer. Previous work on content access in vehicular networks has focused on the current Internet. The techniques include the use of mobility prediction to stage content onto wireless base stations [1], [2], [3], or separate the wireless segment with dedicated transport design from the end-to-end path [4]. All these solutions rely on proxies and overlay-based approaches that are complex and hard to deploy at scale.

Recent years have witnessed a proliferation of Information-Centric Networking (ICN) proposals [5], [6], [7] that provide explicit support for content retrieval in the network layer. – content is retrieved as a sequence of secured data objects (*e.g.,* by using a signature to ensure integrity and authenticity), instead of the traditional solution of establishing an end-to-end byte-stream session. In mobile environment, it is easier to optimize communication at the data object than the byte-stream level: one can potentially fetch objects from a closer location, and "smaller" data objects are more likely to be completed when connectivity is intermittent.

Building on the native benefits of ICN in content delivery, we designed SOFTSTAGE, that performs content staging that uses edge caching to optimize mobile content retrieval as a network layer function, and we implemented it in the XIA future Internet architecture. The design of SOFTSTAGE has the following four features: ① It makes the staging procedure completely transparent to the client application so application developers do not have the need to worry about content retrieval and the complexities from client/content mobility; ② To minimize the management overhead in the edge network, SOFTSTAGE is split into two cooperative parts: Staging Manager on the client side manages the staging policy and also maintains all the associated states, while a stateless and application-agnostic Staging Virtual Network Function (VNF) inside the edge network cooperatively executes content staging; ③ The staging policy for managing the content mobility is based on the philosophy of reactive rather than predictive – we developed a staging algorithm that adapts to the network dynamics to make the staging action "Just-in-Time" rather than blindly excessive/conservative; and ④ We also design a content-aware handoff manager module to minimize the impact of disruptiveness in content delivery during network handoff. Finally, by offloading the network selection and (staging) VNF discovery functions to a service-centric network sensor module, we offer an efficient and flexible interface to allow better management of client mobility and edge VNF discovery.

We evaluate SOFTSTAGE using commodity WiFi device in an indoor mobility testbed with extensive experiments accommodating the real vehicular network conditions, *i.e.,* disconnectivity, dynamic network bandwidth, frequent handoff, high packet loss. We demonstrate that with SOFTSTAGE, a FTP-style application can save 1.5∼10x in downloading a stream of content objects. Note that our design is generic and does not depend on the cause of intermittent connectivity, *e.g.,* be it coverage holes, temporal failure, media contention. While previous work has evaluated staging [3], [8] and heterogeneous network segment management [4], [9] in vehicular networking, SOFTSTAGE integrates them into a single system at the network layer in an application-agnostic manner.

Contributions.

- We design and implement SOFTSTAGE, a content staging network layer function to improve throughput in intermittent (vehicular) networking environments without changing the application semantics and predicting the client mobility.
- We validate the efficacy of SOFTSTAGE by evaluating the

✉: chenren@pku.edu.cn

content download time using both controlled and trace-driven experiments in real testbed to show that SOFTSTAGE achieves up to 10x throughput gain over the FTP-style applications.

II. BACKGROUND AND MOTIVATION

A. Vehicular Content Delivery Challenges

The key characteristic of vehicular networking is the intermittent connectivity that results from handoffs and coverage gaps. For instance, the Carbernet project [4] studied the vehicular Internet access over road-side open WiFi networks in Boston. Their results reported that when a mobile client is moving at urban vehicular speeds, the connection time with APs[1] had a median of 4 sec and mean of 10 sec, and the observed median and mean time between successive encounters was 32 and 126 sec respectively. These number suggest that the time available for content delivery is very limited. Even worse, communication sessions, once established, will often be bottlenecked in the Internet, so useful transmission time over the wireless link is often not used efficiently.

B. Opportunities created by ICN

The goal of Information Centric Networking (ICN) is to optimize information retrieval by making "content" a first class citizen in the network. Specifically, ICN introduces content addresses that represent content objects, *e.g.,* a web page or video segment, instead of specific network devices. Packets sent to a content address are delivered to the nearest device that has the content, and the semantics are that the specified content will be sent to the client that sent the content packet. The primary motivation for this design is that it opens the door for network layer, application-agnostic caching of content, since the network can deliver from the nearest cache or origin server, *e.g.,* as identified by a shortest path routing protocol. One implication is that large objects, such as a video, must be broken up in smaller chunks that can be handled efficiently by caches. Content addresses can be the actual content names [7] or content identifiers derived from the content or its name [10]. As a result, ICNs often can avoid name lookup, which can be expensive [11].

ICNs offer several features that can help optimize vehicular content delivery natively, *i.e.,* at the network layer. First, communication based on chunks, rather than long lived TCP sessions, is more appropriate for environments where the connection time is short, since they can be more likely transferred in their entirety. Second widely deployed network layer caches can naturally be used to stage content closer to the edge, even while the vehicle is disconnected. Finally, the use of edge caches naturally breaks up the long client-server connection in a long Internet segment and a short wireless segment. This makes it possible to fully utilize the wireless segments, possibly bursting chunks as hundreds of $Mbps$, as 802.11ac APs are becoming more pervasive.

[1]Defined as the time interval between when the first beacon and when the last packet was heard from the AP.

In addition to ICN technology, SOFTSTAGE relies on two more features to support efficient, vehicular content retrieval at the network layer.

- **Virtual Network Functions (VNFs)** can be used incrementally deployed in custom network functionality in the network routers and devices. In SOFTSTAGE, we use VNFs to leverage caches in the edge network to stage content (see XCache as a reference implementation introduced in §II-C) and collaboratively work with the clients to process their requests from a closer location. More importantly, *clients can communicate VNFs in edge network different from it is connected to*. This allows clients to optimize content retrieval across multiple caches as they move between edge networks.
- **Multi-homing** makes it possible for applications to use more than one interface to communicate with the wireless infrastructure. Such mechanism can be very useful in separating disjoint functions into different interfaces to reduce the overhead of context switching in the OS and frequency change in WNIC. For instance, wireless clients traditionally use one WiFi interface for both data transfer and network scan. With native multihoming support, the client can use the "data" interface primarily for content retrieval while the "sensor" interface can be used for received signal strength (RSS) based layer 2/3 connectivity selection and VNF discovery.

C. eXpressive Internet Architecture (XIA)

We use XIA [6] as the network architectural basis for SOFTSTAGE. XIA simultaneously supports multiple communication paradigms by using multiple address types. These conclude host identifiers (HID – traditional host-based communication), content identifiers (CID – ICN), and service identifiers (SID – service-centric networking). XIA also supports network identifiers (NID – equivalent to IP prefixes). Taking content retrieval as an example, a data chunk CID, such as a web page or a movie chunk, can potentially be retrieved using different types of addresses. It can be retrieved using its CID (from the nearest location), by contacting a possibly replicated HTTP service (SID), or by contacting the origin server directly (HID located in network ID). Since not all routers must support all address types, XIA allows addresses to contain multiple identifiers, effectively specifying different ways of reaching the destination. To represent fallbacks, XIA uses an address format based on *directed acyclic graphs (DAGs)*. Since SOFTSTAGE only builds on the ICN features of XIA, we will use a simpler representation, namely $CID|NID:HID$. This address means that routers should forward the packet based on CID if they can; otherwise they forward based on $NID:HID$, which is the equivalent of an IP address. By having the fallback ($HID:NID$ in this case), CIDs can be opportunistic in the sense that not every forwarding table has to store all CIDs (or even any). We next introduce the key elements of XIA relevant to SOFTSTAGE.

Elements of Information-centric Networking. The XIA chunk cache, XCache, implements XIA's native ICN support on both end hosts and network appliances (*e.g.,* routers). The

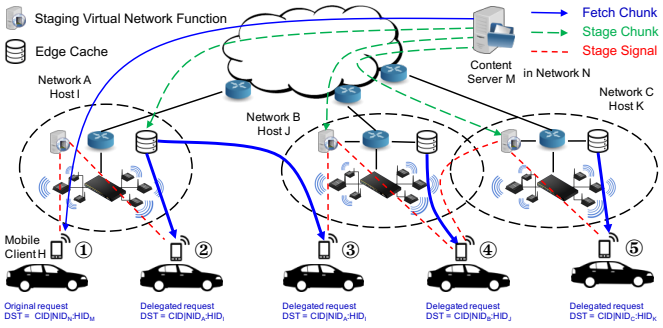


Figure 1. Overview of how SOFTSTAGE benefits vehicular content delivery.

cache stores chunks which it can serve at the network layer when it receives ICN requests, *i.e.,* packets with a CID destination address. Chunks are transferred using a TCP-like reliable protocol connection directly between XCache and the requesting client. Content providers can publish content as chunks in their local XCache, which can serve clients without server involvement. Similarly, clients can optionally store chunks in their XCache for reuse by other clients. XCache on routers can opportunistically cache content that is forwarded by the routers. It can also serve content when the router receives an ICN request, *i.e.,* a packet with a "CID" destination, that is stored in the local XCache. XCache is a network layer module that is tightly coupled to the XIA forwarding engine; it is implemented as a user-level daemon so it can use large caches. For simplicity, the client Staging Manager has been implemented in the client XCache.

Elements of Mobility-centric Networking. XIA natively supports network-layer mobility [12]. Since both HID and SID are the hash of a public key, XIA employs AIP-style accountability (challenge-response protocol) [13] and active session migration [14] to secure client mobility at both host level and session levels. In other words, one can verify the peer's authenticity at any time regardless of its location.

III. SOFTSTAGE

A. System Overview

The goal of SOFTSTAGE is to efficiently use all wireless transmission opportunities to improving content retrieval in highly intermittent networking environments. We use vehicular content delivery as a example and categorize SOFTSTAGE's operation as a sequence of steps as illustrated in Fig. 1: ① The mobile client H initially joins Network A, starts to fetch content objects. Initial chunks are retrieved directly from the server, while the client contacts the edge VNF to stage future chunks into the edge cache. ② For later chunks, H enjoys a smaller network latency by fetching them from the edge cache, and meanwhile maintains the staging session with the edge VNF. ③ After H experiences a coverage gap and joins Network B, it is likely the cache in Network A still holds chunks that have not been fetched by H. Instead of downloading those chunks from the remote server, H continues to fetch the remaining chunks from the cache inside the old network A, while launching the staging procedure within the current network in parallel. ④ With support of multi-homing, H discovers a network with better connectivity and it may handoff to Network C. Specifically, SOFTSTAGE uses the second interface for network discovery rather mobility prediction. This time, H not only performs handoff procedure (similar to ②), but it also contacts the staging VNF in Network C via the current Network B before the handoff procedure. ⑤ H repeats the procedure in phase ②. Note that *the content's ID and its network location is an explicit part of the address used for content retrieval, which is different from a overlay approach and thus application agnostic.*

The aforementioned procedure for content delivery is highly efficient in three aspects. First, the content is directly retrieved from edge networks instead of the original server with lower RTT, thus helping the underlying transport protocol by ramping up faster and getting higher bandwidth. Second, the session benefits from the faster wireless access link, because the Internet can be occasionally slow due to a random bottleneck link. Third, the client is able to concurrently fetch and stage content most of the time, thus minimizing the impact of the intermittent network connectivity.

To actually gain such benefits, XIA offers native support from the network layer – it facilitates accessing and deploying VNF (*i.e.,* XCache) for splitting the traditional end-to-end communication session into a wireless segment and the Internet segment. On top of that, we designed SOFTSTAGE to efficiently orchestrate the resources in both mobile client and edge networks to realize such staging procedure in a transparent, scalable, economical manner.

B. Design Considerations

Application Transparency. Compared to an end-to-end session, the staging mechanism introduces additional tasks, such as deciding when to stage content objects, and managing the process. Although these are important issues, they ought to be transparent to the client application, *i.e.,* application developers should not be involved in managing the content mobility. Therefore, we need a client delegation mechanism to hide all these underlying complexities transparently from the client, *e.g.,* through a simple data-oriented client API.

Distributed State Management. Similar to an end-to-end session, the staging procedure also has state, but in a richer contextual space, *e.g.,* staging progress, chunk (session) membership, and client network location. In addition, the amount of state can grow rapidly as SOFTSTAGE start to serve more clients. Therefore, even though the edge network is expected to be the staging executor, it can suffer from scalability, availability and connectivity issues if it handles this large state space alone. To address this issue, we require the client and the network to cooperatively share the staging work in a client-directed manner: in the control plane, the Staging Manager is placed on the client side that owns the state policy and tracks all the state, because client device is often in a better position of knowing the application requirements and the

contextual information of the networking environment; in the data plane, the Staging VNF inside the edge networks listen to Staging Manager's requests and performs content staging accordingly so it can keep a minimal amount of state.

Content Mobility Management. Content staging is essentially a content mobility management problem, in a sense that content is replicated at a closer device before getting requested. Typical staging techniques are based on predictive methods [3], [8], *i.e.,* assumes client mobility pattern can be accurately modeled and predicted at layer-2, therefore one can stage the "right" content into the right access points (APs) for retrieval. We believe that this approach may not work well in practice. First, APs' availability can be very dynamic (*e.g.,* load balancing, misconfiguration) can dramatically change the temporal-spatial association pattern. Second, for personalized content staging, this process can be too aggressive, *i.e.,* network resource are wasted if too many content objects are early staged but the client decides to quit early and change route for whatever reason. Instead, we adopt a reactive and more economical approach – we stage the content-on-demand in a "Just-in-Time" manner based on the network condition dynamics and the staging execution is performed only after client is associated with the network, to fundamentally address the issues in predictive approach.

Client Mobility Management. Client mobility management refers to the process of switching between access points for better throughput and has been well studied in the context of the current Internet [15], [16], [17] primarily based on received signal strength. However, in a ICN architecture, a session is broken up into a sequence of shorter chunk transfer. This creates the opportunity to time handoffs *between chunk transfer so no transmission is wasted due to interrupted chunk transfers or due to active transport session migration.*

Fault Tolerance. The aforementioned considerations are based on the assumption that all the encountered edge networks have deployed Staging VNF, which may not be the case in practice. A fallback mechanism that allows clients opt to retrieve content objects from the original content publisher is required for robustness enhancement.

C. System Design

The design of SOFTSTAGE involve four components:

- Client Host runs client application (*e.g.,* ftp, video player), which contacts the server application to retrieve the content objects' DAG information and then calls a modified XCache API to fetch the target chunks.
- Server Host runs server application, which listens to the client's request, split the target file into chunks and put them into the local cache for serving the clients.
- Staging Manager runs inside the client host or on a separate device with reliable connectivity with it. It discovers and communicates with Staging VNF, to perform content staging based on its own policy.
- Staging VNF is deployed in edge networks and stages content upon Staging Manager's request.

The functions involved in the whole process are modularized to separate the control from data planes: the control plane handles the client application interface, address management, signaling for staging activity and chunk-aware handoff, as well as managing the content mobility and staging process; the data plane uses the XCache API to fetch chunks from caches and servers across the network. The design of SOFTSTAGE and its operation flow is illustrated in Fig. 2. next, we elaborate on how these modules collectively achieve the goal of SOFTSTAGE.

Staging Manager. As mentioned in §III-B, the design of SOFTSTAGE needs to accommodate application transparency and management of staging state, content mobility and client mobility. We decompose the functions on Staging Manager side into six modules and show their interactions in Fig. 3:

• Chunk Profile populates and maintains the states (Tab. I) indexed by addresses of the content objects (dynamically) registered ❸ by the client. It serves as the database for other modules to update and retrieve information.

• Chunk Manager performs client delegation by providing a delegation API `XfetchChunk*` to client applications for content retrieval with location transparency – it allows fetching the chunk from a closer location (*e.g.,* edge networks) but hides the underlying complexities from the client application. Once called by the client application ❸, `XfetchChunk*` first polls Chunk Profile for the new address (with the fallback path replaced with the NID and HID of the edge network holding the staged/cached content, *e.g.,* $CID|NID_A : HID_I$ in ② of Fig. 1) returned ❻ from Staging VNF. With this new address, it calls the native `XfetchChunk` ❼ to fetch chunks from edge networks if the handoff request flag is not set (to be explained in Handoff Policy). If set, it will clear the flag, signals Handoff Manager to switch network and wait until joining a new network to continue fetching chunks. Finally, it updates Chunk Profile the fetch state of the chunk from NONE to DONE and the latency in between. Note that if Staging VNF is not available in the current edge network (*e.g.,* not deployed, overloaded, or DNS failure), the staging procedure (from ❹ to ❻) will be skipped and the original DAG of the target content object ($CID|NID_N : HID_M$) will be returned. In this case, `XfetchChunk*` directly calls the

Item	Note
Raw DAG	Dest. address with the NID and HID of XCache in the original content server as fallback
New DAG	Dest. address with the NID and HID of XCache in the EdgeNet holding the staged chunk as fallback
Fetch State	BLANK, DONE
Staging State	BLANK, PENDING, READY
Location	The NID and HID of the EdgeNet holding the staged chunk
Fetch RTT	$RTT_{C,EdgeNet}$, round-trip time between client device and EdgeNet
Fetch Latency	$L_{EdgeNet \to C}$, time to fetch one (staged) chunk from the EdgeNet to the client device
Staging Latency	$L_{S \to EdgeNet}$, time to stage one chunk from the original content server to the EdgeNet

Table I
CHUNK PROFILE MANAGES THE KEY INFORMATION FOR CONTENT STAGING.

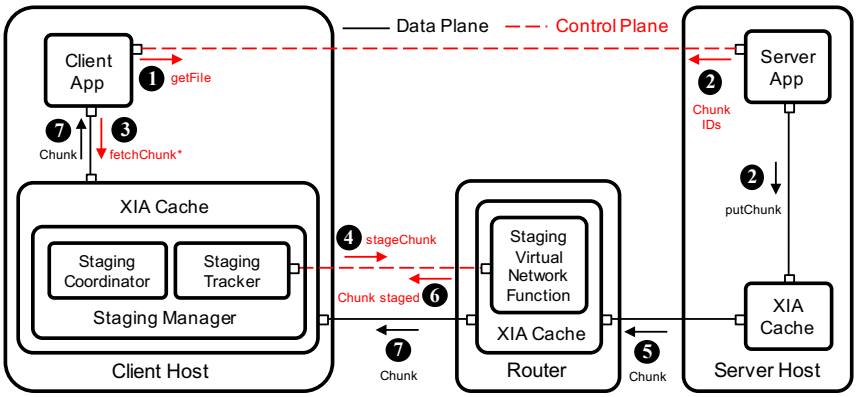


Figure 2. SOFTSTAGE design and its operational flow.

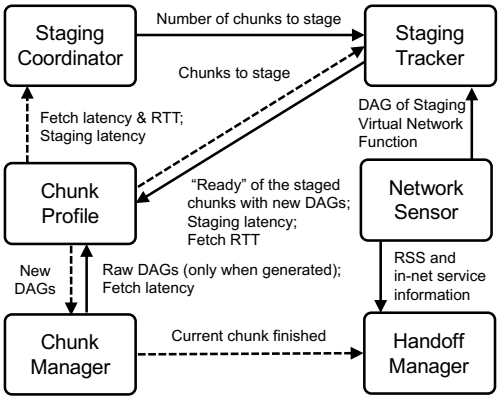


Figure 3. Staging Manager design.

native `XfetchChunk` to fetch the chunk from the original content server and sets the associated staging state to DONE to avoid duplicated staging.

• Network Sensor uses a separate (or virtual) wireless interface to scan for new network connection opportunity, collect RSS measurements, and discover any VNF[2]. This information will be periodically updated to and used by Handoff Manager for better handoff decision. It also sends Staging Tracker the address information of the (staging) service in edge networks.

• Handoff Manager is responsible for deciding when to perform a handoff. In our default policy, we adopt the legacy algorithm that switch to a new AP with a higher RSS than the current one. To further improve the performance by reducing the overhead in active session migration, we developed a *Chunk-aware Handoff* algorithm – it sets the handoff request flag and will not start the handoff procedure until it is informed by Chunk Manager about the completion of the chunk currently being fetched.

• Staging Coordinator polls Chunk Profile and synthesizes all relevant information to decide how many chunk(s) to stage without any assumption of the client mobility pattern, and it then notifies the Staging Tracker. This reactive staging algorithm will be detailed in §III-D.

• Staging Tracker is informed by the Staging Coordinator of the number of chunks to stage, looks up the corresponding addresses from Chunk Profile, forwards them to Staging VNF ❹ and updates their Stage State to PENDING in Chunk Profile. Once receiving the "chunk staged" message back ❻, it updates the Chunk Profile with the address (with EdgeNet's NID and HID), the staging state (to READY), staging latency and fetch RTT of the staged chunks.

Staging Virtual Network Function. It is responsible for prefetching individual chunks and staging them in the XCache for retrieval by the mobile device. This is a very lightweight virtual network function embedded inside XCache that is application-agnostic. It works as follows: first, when requested by Staging Manager, it stages the target chunks from content servers into its local XIA cache; second, it responds to the client by sending the message back including the addresses (with the edge network's NID and HID) and their corresponding latency and RTT to be used by the staging algorithm. Once the chunk is staged, the client can retrieves the chunk locally using the XIA standard `XfetchChunk` API whenever it is connected to the network.

[2]The VNF discovery is achieved by Network Joining Protocol [18] – the access network advertises its presence with any usable VNF information in its beacon message. The detail is omitted here.

D. Staging Algorithm

SOFTSTAGE aims to perform content staging in an economical manner – it stages an "optimal" or minimum number N of content objects ahead of the one currently being fetched into the edge network on a reactive basis, rather than predictively plan for all the remaining ones in the session into different networks/APs beforehand [3], [8]. By doing that, network resources such as network bandwidth and cache space be utilized more efficiently. To ensure the continuity of content fetching on the client side, it is important to keep sufficient chunks staged in edge networks.

Intuitively, N should be selected based on the network state. For instance, it is safer to keep a large N when we have a fast wireless link. Let us denote C as client, S as original content server, $EdgeNet$ as edge network. The corresponding latency and RTT are summarized in Tab. I. Note that in SOFTSTAGE, fetch and staging are asynchronous: Staging VNF can continue to work when the client is disconnected. When clients start to fetch the staged content, Staging VNF will have to catch up very quickly. Since the wireless link can be faster than Internet, it is critical to determine the optimal number of N chunks staged for buffering. In other words, *we want to ensure the time to have the client fetch the staged N chunks is longer than the time it takes to stage a new chunk, otherwise, we need to stage a new chunk immediately*. To stage a new chunk, it takes $RTT_{C,EdgeNet}$ ❹ and ❻ for Staging Manager to query Staging VNF, which in turn takes $L_{S\rightarrow EdgeNet}$ ❺ to stage a chunk from the original server. Similarly, assuming all the chunks are fetched in sequential, it takes $N \cdot L_{EdgeNet\rightarrow C}$ ❼ to fetch all the N staged chunks. Hence, when N satisfies

$$N < \frac{RTT_{C,EdgeNet} + L_{S\rightarrow EdgeNet}}{L_{EdgeNet\rightarrow C}},$$

Design consideration	Responsible module and comments
Application Transparency	The delegation API `XfetchChunk*` coordinates with Chunk Manager to retrieve the DAG for fetching the chunk staged by Staging Tracker
Distributed State Management	Chunk Profile maintains all the state updated by Chunk Manager and Stage Tracker.
Content Mobility Management	Staging Coordinator decides when to stage which chunks based on the network condition (*e.g.,* latency and RTT) estimated by Chunk Manager and Staging Tracker.
Client Mobility Management	Handoff Manager performs chunk-aware handoff based on RSS and VNF information from Network Sensor and fetch state from Chunk Manager.
Fault Tolerance	Chunk Manager will return the chunk's original DAG to the delegation API `XfetchChunk*` if it learns the target chunk was not signaled to stage from Chunk Profile or no Staging VNF available in the edge network from Network Sensor.

Table II
RECAP OF HOW DIFFERENT MODULES IN SOFTSTAGE COPE WITH THE DESIGN CONSIDERATIONS.

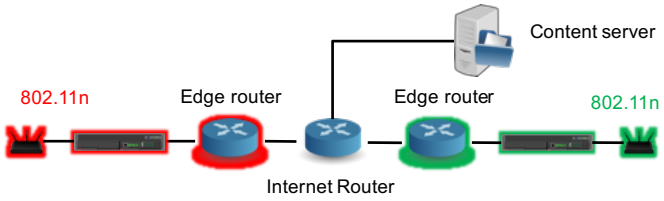


Figure 4. Experimental testbed.

a new chunk needs to be staged immediately.

IV. EVALUATION

We implemented SOFTSTAGE on top of the XIA prototype [10] and used it experimentally evaluate SOFTSTAGE.

A. Experimental Setup

When designing experiments, we should keep in mind that a typical V2I network has three key characteristics: intermittent connectivity, high packet loss rate and networks with overlapping coverage. The first two factors can be captured by real data trace, while the last one can be emulated with a carefully designed network topology and experimental testbed.

Networking Environment Profiling. We employ the dataset from the Carbernet project [4] because it has a variety of V2I traces collected from 124 driving hours and 26000 APs in Boston area, which we believe is extensive and representative. We use 25th, 50th and 75th percentile of the encounter time (from 3 to 12 *sec*), disconnection time (from 8 to 100 *sec*) and packet loss (from 20% to 40%) respectively to emulate different V2I scenarios such as metropolis and urban areas.

Topology and Testbed. We use the network topology shown in Fig. 4 to emulate the V2I networking environment, primarily for supporting inter-network roaming. Specifically, we use COTS WiFi APs [20] rather than hostapd [21], both because the latter's performance is often suboptimal due to software overhead, but also because XIA runs natively on any layer-2 device.

B. XIA Benchmark

To understand the baseline performance of the current XIA software implementation, we first benchmark the throughput of 10 MB file transfers using Xstream (using a XIA byte stream session) and XChunkP (a sequence of 2 MB Chunk XIA transfers), and compare it with that of Linux TCP (based on iPerf [22]) over both a wired and an 802.11n segment. Both XIA byte streams and chunk transfers use the same underlying TCP-like transport protocol. Since XIA is implemented based on the Click modular router [23] as a user-level daemon, its performance is expected to be lower than that of a native Linux TCP implementation.

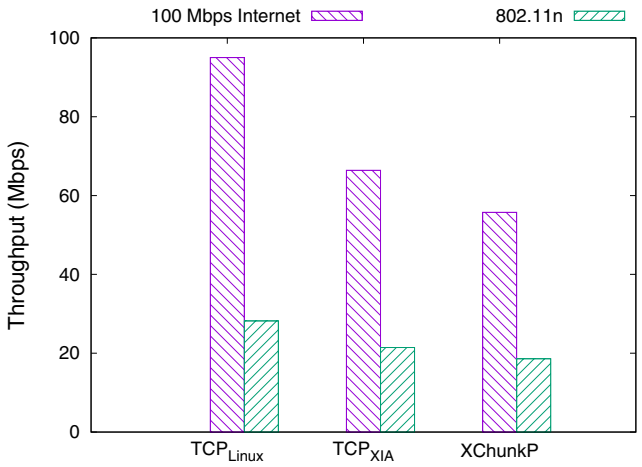
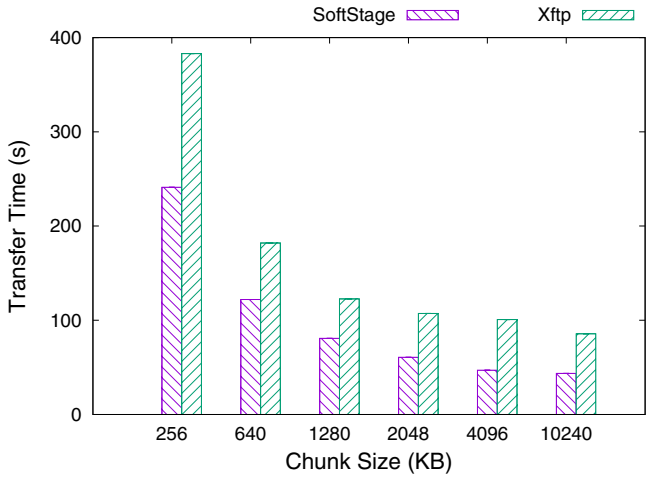


Figure 5. Benchmark results.

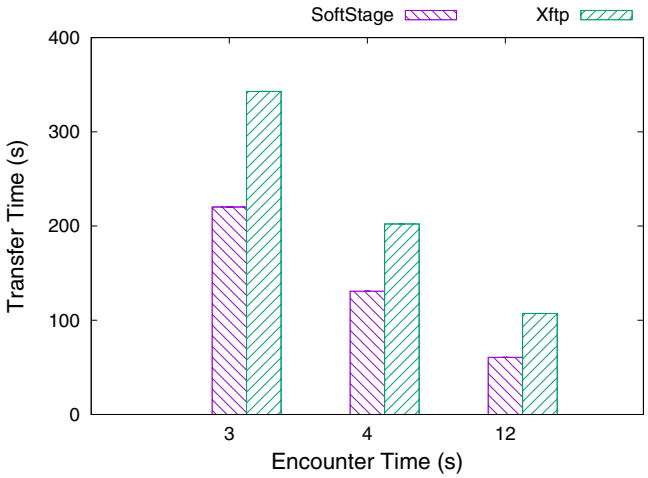
Based on Fig. 5, we can make three key observations. First, in terms of application-level performance, Xstream is comparable to Linux TCP (22 versus 28) over 802.11n, which suggests that the XIA implementation is good enough for wireless experiments in our setting. Second, when running over a wired segment, Xstream is worse than Linux TCP (66 versus 95). However, this is not a problem for our study because it is still faster than the wireless segment and we can change the packet loss rate to emulate different bandwidth on the Internet segment to study its impact on the SOFTSTAGE's performance relative to the fix rate of 802.11. Finally, the performance of XChunkP is slightly worse than Xstream (19 versus 22 and 56 versus 66 on 802.11n and wired segment respectively). This result is expected because the XChunkP transfer is broken up in chunks that are fetched separately and this comes with larger protocol overhead.

C. Controlled Experiments

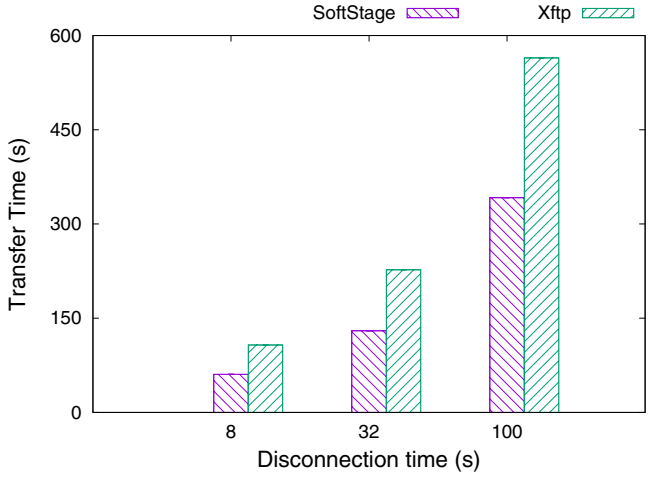
1) Micro-benchmarks: Intuitively, SOFTSTAGE 's behavior depends on the network environments. We summarize the variables of interest in Tab. III and will elaborate on them



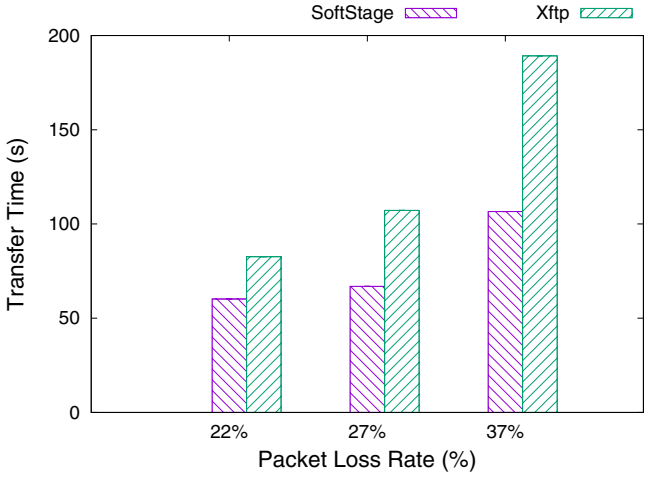
(a) Chunk size



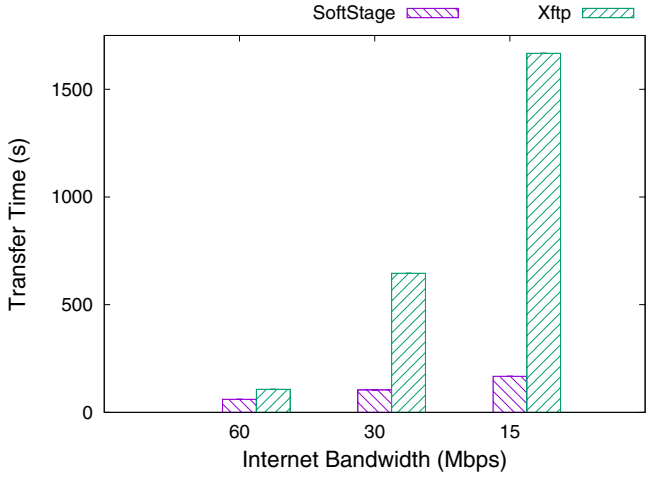
(b) Encounter time



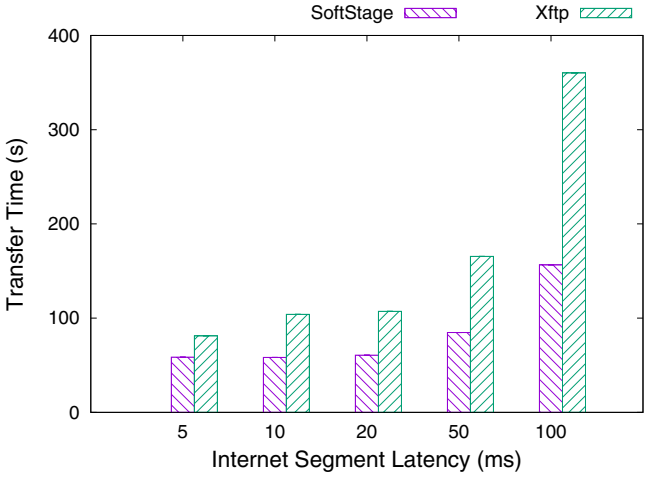
(c) Disconnection time



(d) Packet loss rate



(e) Internet bandwidth



(f) Internet latency

Figure 6. Performance gain of SOFTSTAGE downloading a 64 MB file under different parameters.

as we move on. In this set of micro-benchmark experiments, the mobile client starts to download a 64 MB file and will switch between two edge networks back and forth – it stays *Encounter Time* in each network, and disconnects from it for *Disconnection Time* before joining the other one. We vary the variable of interest and keep the rest of them as "default" to study their impact on SOFTSTAGE's performance.

Application Properties – Chuck Size. The chunk size is directly tied to the application and the QoE. For example, according to the Youtube recommended SDR video bit rates for Standard Frame Rate uploads [19], the chunk size of 0.25, 0.625, 1.25, 2, 4 and 10 MB represent the size of 2 sec' video clip of 360p, 480p, 720p, 1080p, 1440p (2K) and 2160p (4K). Intuitively, bigger chunk size introduces less protocol overhead and hence improves the throughput. But this improvement is not sustainable, simply because it takes more time to complete fetching a chunk. This hurts the performance when fetching the chunk across the network (② in Fig. 1) and taking more time to make the staged chunk ready for use. In Fig. 6(a), we observes this trend for both Xftp and SOFTSTAGE, and SOFTSTAGE consistently outperforms Xftp. SOFTSTAGE achieves 1.59x to 1.96x than Xftp as chunk size increases, because the control plane messages introduce more overhead with smaller chunks.

Client Mobility – Encounter Time. It is jointly determined by client's moving speed and network size (*e.g.,* number of APs aggregated at SSID level). We choose 12-sec, the 75th percentile as our default value because of the recent densification progression of AP deployment by cellular operators – the coverage of such small cells or APs' can be geographically contiguous for a block of street, and this is the number a vehicular client can received WiFi coverage when the traffic is light. We also keep 3 and 4 sec as it represents the case that AP as a single network. In Fig. 6(b), we observe that SOFTSTAGE outperforms 1.77x when encounter time is 12 sec, which is better than 1.55x when there is only 3 sec of network connection. Longer encounter time will reduce the number of chunks experiencing active session migration (a fixed overhead of 1 or 2 sec), and thus more time can be used for content retrieval.

Client Mobility – Disconnection Time. It captures the layer-3 mobility in the context of hard handoff – the total time between client device joins a new network and disconnected from the previous network. We choose 8-sec the 25 percentile as our default value again because of the densification progression in recent years. We also keep 32 and 100 sec for the cases that AP deployment are rather sparse, such as rural area, *etc.*. From Fig. 6(c), we can see that when disconnection happens, SOFTSTAGE enables the edge network to continue staging the remaining partial chunk, so it takes less time to finish fetching the remaining partial chunk when connection is recovered. Since the least disconnection time (8 sec) is long enough for Staging VNF to finish staging, the increasing of disconnection time won't influence the total time that used to fetch chunks. So the performance gain for different disconnection time is close, which is about 1.7x.

Channel Fading – Packet Loss Rate. V2I networking often experiences packet loss because of the large scale fading from static/moving objects' blockage. In this setting, we still choose the number reported from [4] in 2008 since the blockage problem is orthogonal to the recent advance in wireless technology. As shown in Fig. 6(d), when the packet loss increases, the performance gain of SOFTSTAGE goes higher from 1.37x to 1.77x. For SOFTSTAGE, it achieves better results than Xftp because in the case where packet loss cannot be hidden from higher layer (with local link-level retransmission), the packet will be retransmitted from a closer location.

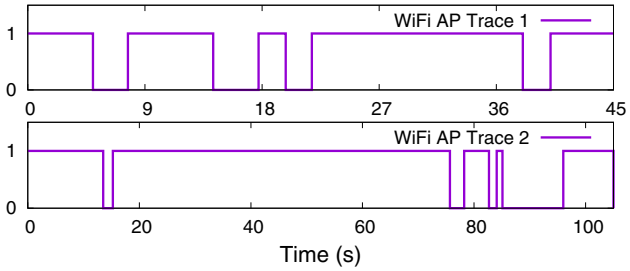
Network Congestion – Internet Bottleneck Bandwidth. Independent from the wireless edge network, the bottleneck bandwidth of the Internet can be affected by many factors. Here, we want to evaluate how it impacts the performance of SOFTSTAGE. Since our testbed is in a controlled networking environment, we can easily emulate and set the Internet bandwidth as 60, 30 and 15 Mbps representing the link speed which is much higher than, higher than and comparable to the wireless by tuning the packet loss rate in the NIC. For Xftp, its throughput is highly dependent on the bottleneck bandwidth between the client and the content server, *i.e.,* the Internet in this study, as shown in Fig. 6(e). However, since SOFTSTAGE monitors the network conditions and will aggressively stage more chunks when the Internet bandwidth is detected slow, especially when the client is disconnected from the network, its performance gain over Xftp boosts from 1.77x to 9.94x when the Internet bandwidth is reduced from 60 to 15 Mbps.

Parameter	Default	Note	Others candidate values
Chunk Size	2 MB	2 secs' 720p Youtube video clip [19]	0.25, 0.625, 1.25, 4 and 10 MB
Encounter Time	12 sec	Theoretical maximum duration associated with the same SSID[a]	3 and 4 sec
Disconnection Time	8 sec	Time between two consecutive encounters	32 and 100 sec
Packet Loss Rate	27%	Wardriving measurements in vehicular content delivery [4]	22% and 37%
Internet Bandwidth	60 Mbps	Typical bottleneck bandwidth in WAN with moderate congestion[b]	15, 30 Mbps
Internet Latency	20 ms	Typical RTT to CDN (*e.g.,* web portals, streaming media, *etc.*)	5, 10, 50 and 100 ms

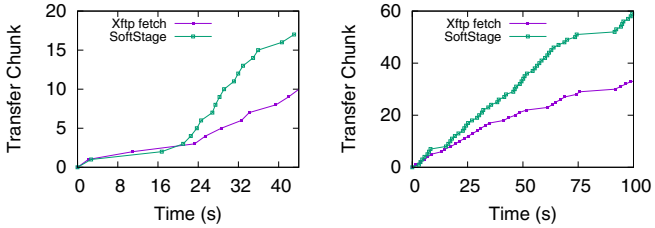
[a]The interval between hearing the first beacon and the time at which the last beacon was heard from the same AP. Therefore this approximation assumes the overhead of layer-2 mobility such as (re)association and authentication can be optimized to near-zero by agent/proxy and mobility controller.

[b]The measured maximum application level throughput the current XIA transport implementation can achieve over a wired segment without introducing any extra latency. This number can even be much higher in reality.

Table III
PARAMETER SETTING FOR EXPERIMENTS.



(a) Trace Shape



(b) Results for trace 1 and 2

Figure 7. Trace-driven experiments.

Content Location – Internet Latency. The round trip time across the Internet can be affected by queuing delay, load balancing and physical location of the CDN server. We add the latency in the NIC on the content server to capture their overall effect. As shown in Fig. 6(f), we observe that the performance gain of SOFTSTAGE increases from 1.38x to 2.3x as the Internet segment latency or RTT increases from 5 ms to 100 ms. The reason is similar to our analysis on the Internet Bottleneck Bandwidth – when the Internet is detected slow, SOFTSTAGE will take advantage of the disconnection period to aggressively stage more chunks earlier to a closer location.

D. Handoff Policy

Handoff are frequent in vehicular networking. All the previous experiments focus on the hard handoff – the client never experiences the case that multiple networks' coverage overlaps, which brings the opportunity for SOFTSTAGE to perform step in phase ④ in Fig. 1. In this context, interestingly, the timing of handoff can also have an impact on performance. Here, we study two handoff policies: *Default.* Blindly switches to the network with a stronger received signal strength; *Content-aware.* Switches to new network until finishing fetching the current chunk. Note that the Staging VNF in new network receives stage signal and performs staging before the handoff procedure. Our experiments use the default setting in previous experiments where the encounter time of each network is 12 sec, while the overlap between two networks are 3 sec. Our results show that our proposed content-aware handoff policy can reduce 21.7% download time in comparison to the default policy. We expect the performance gain will be more significant when the active session migration takes longer time, in the situations when the edge networks or the wireless channel is more congested.

E. Trace-driven Mobile Experiments

In order to evaluate the performance of SOFTSTAGE in real vehicular networking environments, we conduct a day-long wardriving across multiple street blocks in popular areas in Beijing, and only record the trace from the APs deployed by cellular operators. The reason is because in terms of legal access, this is more realistic to assume they are accessible with predictable performance, in contrast to the private APs owned by local residents or enterprise. The collected WiFi traces go to two extremes: network coverage either reaches above 80%, or less than 2%, and we select two traces from the former category since that it is more reasonable to use cellular networks for Internet access than in the latter case. As plotted in Fig. 7(a), we use 1 and 0 in Y axis to represent the period when the device is connected to the network or not. We choose two traces with different network connectivity patterns. As shown in Fig. 7(b), with SOFTSTAGE, the mobile client can download almost twice the content objects in the same networking environment.

V. DISCUSSION

Extension to Video Streaming and Web. Although in this paper SOFTSTAGE only demonstrates the benefits it can bring to a ftp-like application with fixed chunk size and content, it actually provides native support for general content delivery, such as rate adaptive Video-on-Demand (VoD) [24] and dynamic object in Web applications [25].

ICN and CDN. We note that CDN presents the mainstream solution for reducing content retrieval. Yet, our approach is complementary to rather than incompatible with CDN – with caching at the network layer, one can deploy edge VNF like SOFTSTAGE on top of ICN to improve personalized content delivery in pervasive layer-3 devices rather than proxy-based solutions.

Content Cache Management Policy. We only consider one content cache per access network, which has significant limitations in latency, availability, fault-tolerance and scalability. We plan to explore different caching policies, load balancing strategies to address these problems in the future.

VI. RELATED WORK

Content Staging/Prefetching in Vehicular Content Delivery. This topic has been widely studied in the past years. To our best knowledge, most of them use predictive methods [1], [2], [3] based on human mobility modeling [26], [27], [28]. The work [1] develops a staging strategy considering a client's rich mobility information (location, direction and speed) as well as different bandwidth of the edge networks. In [29], a technique has been described to use the cellular network to send data prefetch requests and the data is staged at the prefetch agents located at hot-spotted networks. The work [3] predict mobility and connectivity from people's drives and develop scripted handoff and data transfer strategies to speed up download performance. Our work differ from them in that we argue

human mobility can not be always accurately predicted and instead choose a reactive approach to stage the content-on-demand for Vehicular WiFi access. SOFTSTAGE works in the context that client chooses the network for connection without the knowledge of mobility prediction.

Vehicular Content Delivery in Future Internet Architecture. NDN [7] allows applications to use data name for content retrieval from any content store inside the networks. On top of that, Navigo [30] developed solutions to the problem of mapping data names to data locations and forwarding Interest packets along the best path, and also designed an adaptive discovery and selection mechanism that can identify the available data sources across multiple geographic areas and can quickly react to sudden changes in vehicle networks. MobilityFirst [31] is unique in its GNRS, a distributed naming service system that maintains the mapping of GUID and network address. It enables the edge networks to effectively collect the client network association trace. EdgeBuffer [8] leverages this information to predict the spatial and temporal trajectory of each client, estimates the time intervals in which the clients will be covered by each AP and predictively stage the content into the APs. SOFTSTAGE differs from it in that its staging behavior is reactive and thus more robust to the dynamic client mobility pattern.

VII. CONCLUSION

Staging is an invaluable mechanism for improving the downlink throughput and performance of mobile applications. In comparison with prefetching, it is cheaper and is becoming more deployable due to rise of mobile edge computing where massive cache and replicated VNFs will be pushed to the edge networks. We demonstrate that it is feasible to design and implement a network layer function named SOFTSTAGE, that can effectively leverages the edge resources to perform content staging, without any assumption about the client mobility pattern. SOFTSTAGE also presents a decent effort to synthesize and incorporate the efficient network layer mobility and ICN features to manage the heterogeneous network segments and provide a practical solution for optimizing the content delivery in intermittent network connectivity.

ACKNOWLEDGMENTS

This work is supported in part by National Key Research and Development Plan, China (Grant No. 2016YFB1001200), National Natural Science Foundation of China (Grant No. 61802007 and 61672499) and Science and Technology Innovation Project of Foshan City, China (Grant No. 2015IT100095).

REFERENCES

[1] Tao Ye, H Jacobsen, and Randy Katz. Mobile awareness in a wide area wireless network of info-stations. In *ACM MobiCom*, 1998.

[2] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. Enhancing interactive web applications in hybrid networks. In *ACM MobiCom*, 2008.

[3] Pralhad Deshpande, Anand Kashyap, Chul Sung, and Samir R Das. Predictive methods for improved vehicular wifi access. In *ACM MobiSys*, 2009.

[4] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: vehicular content delivery using wifi. In *ACM MobiCom*, 2008.

[5] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *ACM CoNEXT*, 2009.

[6] Dongsu Han, Ashok Anand, Fahad Dogar, Boyan Li, Hyeontaek Lim, Michel Machado, Arvind Mukundan, Wenfei Wu, Aditya Akella, David G Andersen, et al. Xia: Efficient support for evolvable internetworking. In *USENIX NSDI*, 2012.

[7] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3), 2014.

[8] Feixiong Zhang, Chenren Xu, Yanyong Zhang, KK Ramakrishnan, Shreyasee Mukherjee, Roy Yates, and Thu Nguyen. Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture. In *IEEE WoWMoM*, 2015.

[9] Fahad R Dogar and Peter Steenkiste. Architecting for edge diversity: supporting rich services over an unbundled transport. In *ACM CoNEXT*, 2012.

[10] eXpressive Internet Architecture. https://github.com/XIA-Project/xia-core/wiki.

[11] Ankit Singla, Balakrishnan Chandrasekaran, P Godfrey, and Bruce Maggs. The internet at the speed of light. In *ACM HotNets*, 2014.

[12] David Naylor, Matthew K Mukerjee, Patrick Agyapong, Robert Grandl, Ruogu Kang, Michel Machado, Stephanie Brown, Cody Doucette, Hsu-Chun Hsiao, Dongsu Han, et al. Xia: architecting a more trustworthy and evolvable internet. *ACM SIGCOMM Computer Communication Review*, 44(3), 2014.

[13] David G Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable internet protocol (aip). In *ACM SIGCOMM*, 2008.

[14] Alex C Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In *ACM MobiCom*, 2000.

[15] Sudarshan Vasudevan, Konstantina Papagiannaki, Christophe Diot, Jim Kurose, and Don Towsley. Facilitating access point selection in ieee 802.11 wireless networks. In *ACM IMC*, 2005.

[16] Ishwar Ramani and Stefan Savage. Syncscan: practical fast handoff for 802.11 infrastructure networks. In *IEEE INFOCOM*, 2005.

[17] Anthony J Nicholson, Yatin Chawathe, Mike Y Chen, Brian D Noble, and David Wetherall. Improved access point selection. In *ACM MobiSys*, 2006.

[18] Network Joining Protocol. https://github.com/XIA-Project/xia-core/wiki/NetJoin.

[19] https://support.google.com/youtube/answer/1722171?hl=en.

[20] Tl-ap1750c-poe. http://www.tp-link.com.cn/product_422.html.

[21] Hostapd. https://w1.fi/hostapd/.

[22] UDP iPerf The ultimate speed test tool for TCP and SCTP. https://iperf.fr/.

[23] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3), 2000.

[24] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *ACM SIGCOMM*, 2015.

[25] Michael Butkiewicz, Daimeng Wang, Zhe Wu, Harsha V Madhyastha, and Vyas Sekar. Klotski: Reprioritizing web content to improve user experience on mobile devices. In *USENIX NSDI*, 2015.

[26] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive wi-fi mobility data. In *IEEE INFOCOM*, 2004.

[27] Jungkeun Yoon, Brian D Noble, Mingyan Liu, and Minkyong Kim. Building realistic mobility models from coarse-grained traces. In *ACM MobiSys*, 2006.

[28] Anthony J Nicholson and Brian D Noble. Breadcrumbs: forecasting mobile connectivity. In *ACM MobiCom*, 2008.

[29] Naoki Imai, Hiroyuki Morikawa, and Tomonori Aoya. Prefetching architecture for hot-spotted networks. In *IEEE ICC*, 2001.

[30] Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, and Serge Fdida. Navigo: Interest forwarding by geolocations in vehicular named data networking. In *IEEE WoWMoM*, 2015.

[31] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(3), 2012.