# NETWORK-ASSISTED MULTIHOMING FOR EMERGING HETEROGENEOUS WIRELESS ACCESS SCENARIOS

**BY SHREYASEE MUKHERJEE**

**A thesis submitted to the**

**Graduate School—New Brunswick**

**Rutgers, The State University of New Jersey**

**in partial fulfillment of the requirements**

**for the degree of**

**Master of Science**

**Graduate Program in Electrical and Computer Engineering**

**Written under the direction of**

**Professor Dipankar Raychaudhuri**

**and approved by**

_____

_____

_____

**New Brunswick, New Jersey**

**January, 2014**

## ABSTRACT OF THE THESIS

# Network-Assisted Multihoming for Emerging Heterogeneous Wireless Access Scenarios

**by Shreyasee Mukherjee**

**Thesis Director: Professor Dipankar Raychaudhuri**

This thesis presents a technique for enabling multihoming in the emerging heterogeneous ("hetnet") mobile wireless access scenarios, based on the MobilityFirst Future Internet Architecture. Most mobile devices now have dual wireless interfaces (such as Wi-Fi and LTE) and the proposed technique can use either or both to achieve significant improvements in performance and service quality. In particular, our approach shifts the burden of policy expression and data-striping from end-nodes to in-network nodes, and utilizes named object routing with GUIDs to establish multiple paths to destination mobile devices. The proposed multihoming technique uses hop-by-hop backpressure for data striping at the bifurcation router and includes a robust mechanism to reduce reordering of packets at the receive buffer. We quantify the performance gains using detailed NS3 based simulations and present results from a thorough parametric study to determine the effects of data-rate, delay and hop-count difference between multiple available paths. Finally, we show that when multiple interfaces are available, simultaneous use of both the interfaces is beneficial only under certain conditions depending on the ratio of the data-rate of the interfaces and the size of the flow.

# Acknowledgements

I would like to express my deepest gratitude to my adviser, Prof. Dipankar Raychaudhuri for his support and invaluable guidance. His constant encouragement, sound advice and patience with correcting and re-correcting the initial drafts has helped shape up most of this work. I am also highly grateful to Ivan Seskar and Kiran Nagaraja, for their feedbacks and for always having an answer to every doubt I approached them with. I would like to thank Prof. Roy Yates and Prof. Yanyong Zhang for being on my thesis committee and providing invaluable insights during weekly meetings. I probably cannot thank my brother enough, for being by my side no matter what. All my friends at WINLAB also deserve special mention. They provided moral support and encouragement and often went out of their ways to help me out. Finally, I would like to thank my friend Akash Baid, for all the discussions that we had every single day and the mentorship that he provided throughout, without which this work would not have been possible.

# Dedication

To my mother for her untiring encouragement and love

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

While the basic design of the Internet has remained largely the same since its inception, the manner in which devices connect to it has seen a dramatic change over the last decade. This change - from fixed, wired access to predominantly wireless access over Wi-Fi and 2G/3G/4G cellular technologies has led to several recent efforts towards a clean slate re-design of the Internet architecture [1, 2, 3]. As these works have shown, efficient and flexible support of mobile devices may require a fundamental rethinking of the underlying routing and transport mechanisms. In particular, the network protocols need to be redesigned to support intrinsic wireless access network properties such as device mobility, varying link quality, disconnection and multi-network access. It is noted that while wired end-points typically have just a single network interface, most wireless devices such as smartphones or laptops are equipped with 3-4 radio interfaces such as LTE, 3G, Wi-Fi and Bluetooth, making multi-network access a mainstream option for the majority of mobile users rather than a specialized scenario.

While devices with multiple interfaces are increasingly common (for example almost all smartphones have both Wi-Fi and 2G/3G/4G radio front-ends), the existing methods for utilizing these interfaces in a sequential manner do not take full advantage of parallel connectivity opportunities to improve users' quality-of-experience. In particular, conventional 'hetnet' access mechanisms simply connect the device through Wi-Fi whenever an access point is in-range and defaults to the cellular connection otherwise. Simultaneous access through both these interfaces can vastly improve the bit-rate and availability compared to one-interface at a time in such scenarios. The manner in which multiple interfaces should be used also depends on the application requirements, which vary from application to application, as shown in Table. 1.1. For example, video

| Application Type | Network Stack and Architectural Requirements | |
|---|---|---|
| | **Meta-level Service** | **Transport Service** |
| Peer-to-peer data transfer | Two-way unicast/broadcast | Moderate latency |
| Web browsing | Two-way unicast | Low latency, smaller size flows |
| Video streaming | Dynamic multicast/unicast | QoS aware, low latency |
| Sensor data upload and app updates | Contextual retrieval | Delayed delivery |
| Common Requirements | Robust mobility management, Device multi-homing support, Disconnection/delay tolerance, Congestion control | |

Table 1.1: Networking Requirements for Different Applications

playback, which requires high reliability, can benefit from the same packets being sent through all available interfaces, while fast file download applications can make use of higher bandwidth that comes through striping a packet stream for interlaced transfer over multiple interfaces. Reference [4] motivates the readers to understand the need for multihoming in current wireless access scenarios by providing seven application specific use-cases.

## 1.1 Problem Statement

In this thesis, we describe a novel network-assisted approach for supporting multi-homed devices in MobilityFirst [1], a specific name-based Internet architecture, however the described approach is applicable to other name-based architectures as well. In MobilityFirst, the permanent host-identifiers are called Globally Unique Identifiers (GUIDs), which are dynamically mapped to the network addresses (NAs) through a centralized mapping service called the Global Name Resolution Service (GNRS). Our multihoming approach makes use of network-assistance in two important aspects. First, the GNRS is used by multi-homed nodes to specify the availability of multiple interfaces and the corresponding preference policies on how to use the interfaces. Second, the task of data-striping is shifted from the end-host stack to the in-network routers which have a better view of the network. Specifically, our in-network data-striping algorithm makes use of hop-by-hop backpressure for determining path capacities of links as shown in

Fig. 1.1 and described in detail in Chapter 3.

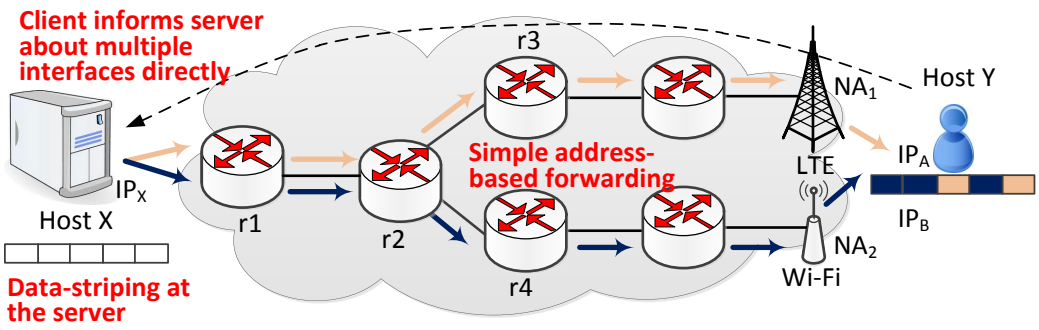
## 1.2 Contributions

The key contributions of this thesis are as follows:

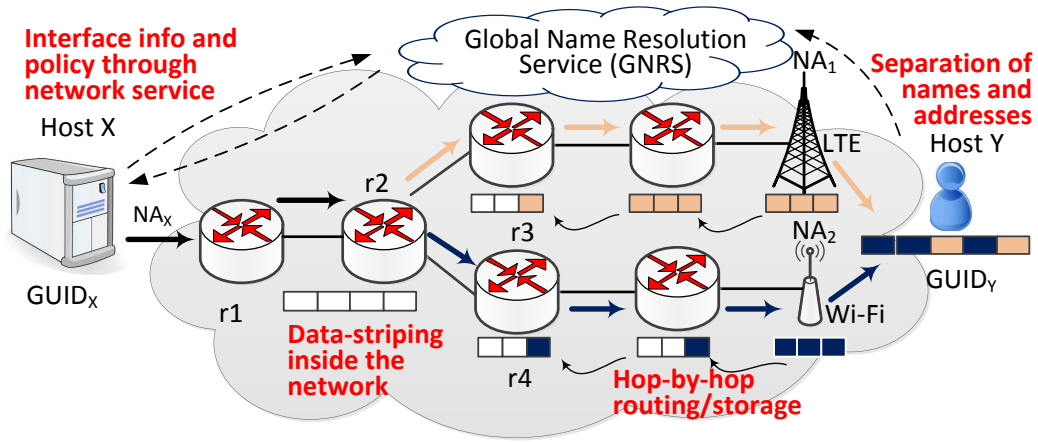
- A flexible, network-assisted approach to support multi-homed devices in the Internet has been described. Several modes of utilizing multiple interfaces (e.g. unicast to most suitable interface, simultaneous transfer through all interfaces, etc.) is enabled through this approach.
- A specific in-network data-striping technique has been proposed, that relies on per-hop backpressure for splitting flows amongst different paths.
- Evaluation results from detailed NS3 simulations have been provided, which show additive throughput gains for multihomed devices.
- The effect of network heterogeneity on the striping algorithm has been analyzed and an optimized technique to intelligently stripe has been introduced, so as to effectively minimize the amount of reordering and consequently reorder buffer requirements at the receiver side.

## 1.3 Related Works

Proposals on introducing multihoming in the Internet can be categorized based on the protocol layer at which they argue the support for multihoming should reside in. Some of the earliest works in this area, such as [5, 6], targeted the link layer for aggregating bandwidth over homogeneous links. However to make it more generally applicable in heterogeneous settings, later works, such as [7, 8], propose network layer mechanisms for multihoming. Reference [7] uses an IP-in-IP tunneling mechanism, but assumes that the link quality remains constant for the lifetime of a flow. In comparison, [8] proposes a network proxy scheme for bandwidth aggregation, but requires fine-grained feedback from the host to the proxy. FatVAP [9] and MultiNet [10] propose virtualizing




(a) End-to-end Multihoming in IP based networks




(b) Network-assisted Multihoming in name-based networks

Figure 1.1: Key conceptual differences between conventional TCP/IP based multihoming and the proposed network-assisted approach

the network layer and allow fast switching between multiple connections, without striping for better utilizing available WiFi connections. In contrast, Horde [11] aggregates multiple cellular connections from different ISPs. Building on top of Horde, authors in [12] propose Tavarua, that perform network layer striping for a video streaming application. Moving upwards in the protocol stack, most recent efforts have gone towards a transport layer approach to multihoming support [13, 14, 15, 16, 17]. While these end-to-end transport-layer proposals for multihoming have started seeing some limited deployments, they offer limited flexibility in the manner in which the multiple interfaces can be used. For example specifying and switching between the replication-mode

and the aggregation-mode needed for the video playback and file download applications mentioned before, requires complicated overlay hacks. In this regard, Multipath TCP [18] allows multipath aware applications to express preference policies for multihomed devices. MAR, a custom router infrastructure has been prototyped [19], that allows any of such transport layer mechanism to interact with its session layer and perform striping or switch interfaces on detection of bottleneck. While this works well when the end-to-end link qualities are relatively stable, several studies have shown the inefficiencies of the underlying TCP protocol itself when the path contains wireless links characterized by varying link qualities [20]. Finally there has also been a considerable amount of work on application layer striping, such as earlier works using XTFP, a modified FTP protocol [21] and more recently [22], the later being applicable only for stationary hosts and requires application specific optimization, that is not salable as such, considering the diverse set of applications that could run on an end node.

Fig. 1.1(a) shows the general philosophy of aggregating bandwidth in the realm of end-to-end transport - an intelligent data-striping layer at the sending end-host divides the flow according to its estimate of the available bandwidth along the two paths, and the received packets are combined at the receiving end-host through a shim re-ordering layer. In contrast, we propose a network-assisted approach shown in Fig. 1.1(b), a transport-layer solution for multihoming, but one in which the end-node stacks are relieved of the policy expression and data-striping tasks.

## 1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides a brief overview of the MobilityFirst architecture, based on which our algorithm is explained in Chapter 3. In Chapter 4 we present detailed simulation results and finally conclude the thesis in Chapter 5 with a discussion on future works.

# Chapter 2

# Overview of the MobilityFirst Architecture

The MobilityFirst architecture is a clean-slate design of the Internet, founded on the premise that in the near future, the number of mobile devices will far outnumber the number of stationary hosts/servers for which the Internet was initially designed for [23]. The design goals, key architectural concepts, protocol details, and prototyping efforts are described in detail in the earlier works [24, 25, 26, 27]. In this section, we walk-through the scenario shown in 2.1 to explain the basic features of MobilityFirst and the design principles that enable network-assisted multihoming.

## 2.1 Naming and Name Resolution

The MobilityFirst architecture is built upon a new name-based service layer which serves as the "narrow-waist" of the protocol stack. The name-based service layer uses flat globally unique identifiers (GUIDs). GUIDs are different from the IP addresses of the current Internet architecture in two significant ways: (i) IP addresses are overloaded to signify both the identity and the location of an end-point, whereas GUIDs serve just as the long-lasting, consistent identifiers, (ii) IP addresses are typically assigned to net devices, but GUID is a single abstraction that covers a broad range of communicating objects - from a simple device such as a smartphone, a person, a vehicle, a group of vehicles, a piece of content, and even context. GUIDs are assigned by one of multiple Name Certification Services (NCSs) and is derived through a cryptographic hash of the public key corresponding to that object. The GUID being directly derived from the public key gives it a self-certifying property, i.e. authenticating a node does not require an external authority [28]. GUIDs are dynamically mapped to a set of network addresses (NAs) or locators corresponding to the current points of attachment of the

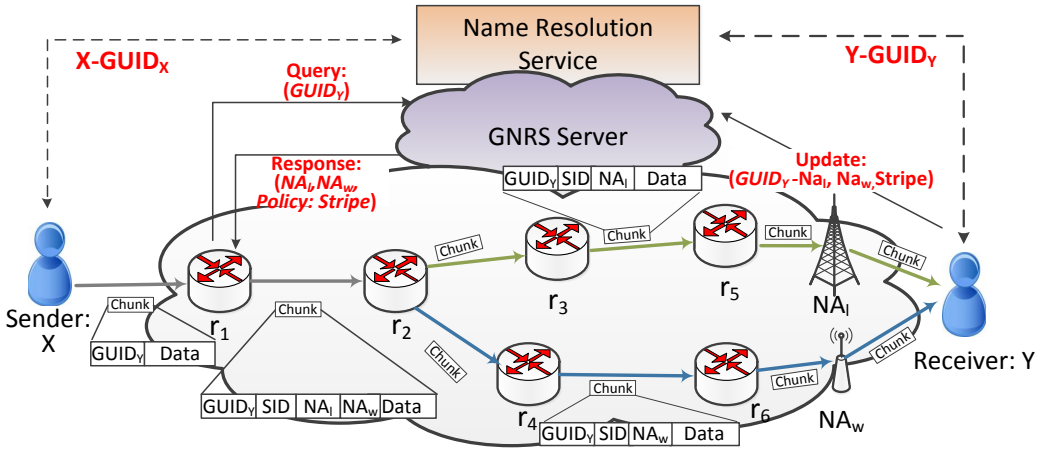



Figure 2.1: Example scenario for data transfer to a multi-homed client in the MobilityFirst GUID routing framework

network object to the Internet through a logically centralized but physically distributed Global Name Resolution Service (GNRS) as shown in Figure 2.2 (see [27] for design, implementation and performance details about GNRS). This enables a scalable name-based service API, i.e., packets can be sent based on the GUID of the destination, which is automatically resolved to the current NA or NAs based on where in the network the object is located.

Fig. 2.3 illustrates how early, progressive,late and re-binding could help in mobility and interface management of a destination GUID. Concretely, in early binding, the destination GUID is mapped to the corresponding NA(s) at the source-attached router, and henceforth data is routed through the network based on that NA. This has the advantage of reducing the GNRS lookup delays for consequent hops and is justified, if the destination does not change its attachment point(s) during the lifetime of the flow. In the case of progressive binding, the destination GUID is mapped to the NA(s) at every hop, and thus, while, on one hand, it allows the chunk to always have up-to-date information of the destination's point(s) of attachment, it experiences more end-to-end delay due to GNRS lookup at every hop. Late binding does slightly better latency management, by incurring lookup delays at every hop, till some intermediate router in the network, where it binds the address to the GUID and does does no further lookups

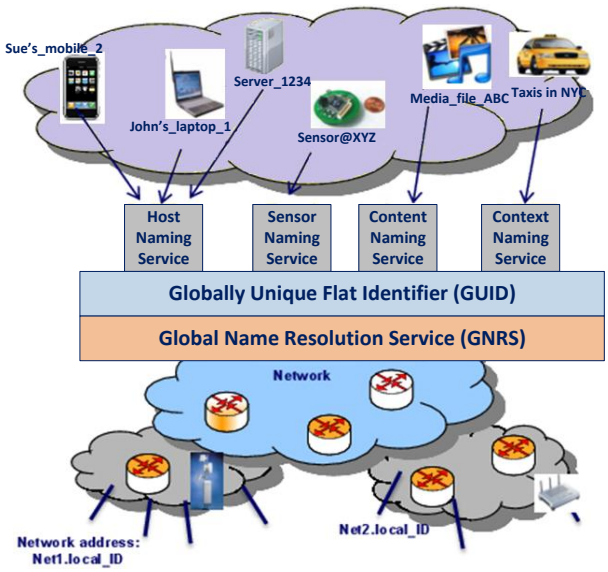



Figure 2.2: Separation of Identification and Network Location in the MobilityFirst Architecture

downstream of the binding router. For the purpose of our evaluation, we however try to achieve a compromise between the no-lookup at the intermediate hops (early-binding) and lookups at every hop (progressive binding), by having an early-binding at the start, and re-binding only if the next hop is not available due to temporary disconnection, link failure or end host mobility. This helps in keeping the end-to-end lookup delay within acceptable limits but might cause rerouting of some packets if the destination changes its point of association while those packets are in flight. It is important to note that the entire flow would not need to be re-routed as packets pending at the source would automatically be routed to the correct NA(s) following an early binding.

In Fig. 1(b), when host Y connects to the internet, it is assigned $\text{GUID}_Y$ , by one of the multiple NCSs, which is mapped to the set of network addresses corresponding to $Y$s current points of attachment (NA1 and NA2). This dynamic mapping of GUIDs to NAs is initiated by Y sending an update message to the GNRS. Y also has the flexibility of expressing its preference policy (for e.g stripe through all, only Wi-Fi, only LTE, replicated data through all, etc.) through this update message. When another host X wishes to send data to Y, it pushes the data out addressed to $\text{GUID}_Y$ . Router

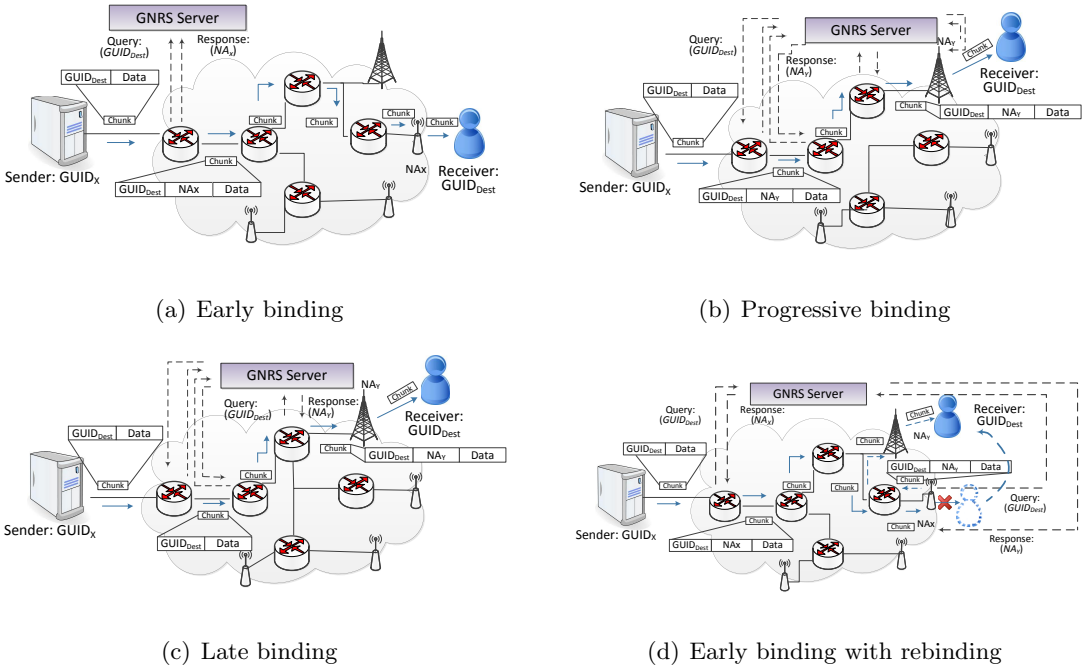



Figure 2.3: Conceptual illustration of mechanisms of GUID to NA binding

r1, issues a GNRS query to obtain the up-to date attachment points along with the user-expressed policy, if any, and forwards the data accordingly. This separation of naming and addressing inherently enables seamless use of multiple interfaces, since every time $GUID_Y$ changes a network attachment point, it updates the corresponding NA value in the GNRS, and in-network routers can query the GNRS to obtain these up-to-date locators.

## 2.2 GSTAR Routing Protocol

At the routing layer, MobilityFirst runs Generalized Storage Aware Routing (GSTAR) protocol, which provides complete visibility of the network in the intra-domain level [24], in conjunction with the Edge-Aware Inter-domain Routing Routing (EIR) protocol, that provides enhanced information about network topology and edge network properties [29]. GSTAR unifies key techniques from mobile ad-hoc networks (MANET) and disruption-tolerant networks (DTN) routing protocols, which makes it particularly suitable for mobile nodes. In GSTAR, all nodes periodically broadcast fine grained link quality information, in the form of flooded link state advertisements(FLSAs), that

contain the short term and long term Expected Transmission Time(SETT and LETT) of their current 1-hop neighbors. These are then used to calculate the overall path quality to all other nodes using Dijkstra's shortest path algorithm [26]. Path selection and transmission decisions are based on factors such as link availability and link quality in terms of the ratio of SETT over LETT. At the inter-domain level, this fine grained information is disseminated in a telescopic fashion, such that nodes farther away have coarser-grained information, while still having some level of visibility of the network. EIR also allows ISPs to have the flexibility to expose optional network properties such as bandwidth, availability, variability, etc.

Note that the decision of whether to stripe at every hop is taken based on the next-hop for the end-host attachment points in the Internet and as long as the routing algorithm provides some visibility on determining what the next hop could be for a given NA, the point of bifurcation could be determined. As such, it is equally applicable when a different inter- or intra-domain routing protocol is used, including the current de-factos: BGP and IS-IS.

## 2.3 Hop-by-Hop Transport Protocol

Since in this work, we enhance the underlying transport layer to enable support for multihomed nodes, we explain the transport protocol used in MobilityFirst - Hop [30] in some detail. Hop is a clean slate transport protocol that is intrinsically different from TCP in three respects. Firstly, instead of packets, routers transfer chunks which are large blocks of concatenate packets. Before sending a chunk to its next hop, a sender sends a control message *CSYN*, on receipt of which the receiver sends a *CACK*, which contains a bitmap of the packets of the chunk that it has correctly received. Secondly, routers utilize in-network caching to temporarily cache in-transit chunks and reduce re-transmission overhead making the setup more robust to intermittent disconnections. Finally, routers use hop-by-hop backpressure through an ack-withholding mechanism, wherein every router monitors the difference between<the number of received chunks for a source/destination pair>and<the number of chunks successfully transmitted to its downstream hop>and limits it to a maximum value $H$. Once, a router has $H$ pending

chunks, it stops sending *CACKs* to its upstream neighbor for newer chunks of the same flow. As explained in [30], unlike end-to-end feedback which could be error-prone, this per-flow feedback mechanism is more robust and provides better utilization of resources at no additional overhead.

# Chapter 3

# Baseline Multihoming Algorithm in the MobilityFirst Architecture

The key features of MobilityFirst show that it can natively support multiple points of attachments of an object to the internet in terms of naming and routing capabilities. In this chapter, we highlight how these features are utilized to implement two types of multihoming applications, namely, 1) the best interface data transfer and 2) interleaved data through multiple interfaces.

## 3.1 Best Interface Data Transfer

In conventional hetnet access mechanisms, a user might be connecting to the internet through Wi-Fi whenever an open access point is in-range and default to the cellular connection otherwise due to current data usage caps by cellular network providers. MobilityFirst provides the flexibility for an application or user to express its policy in the GNRS and as such supports such legacy applications which allow use of only one interface at a time. (The user-defined policy is expressed and forwarded through an optional SID field in the chunk header, similar to the ToS field in an IP header). Even if the policy is single interface or only Wi-Fi, or only LTE, MobiltyFirst aims to reduce switching overhead, such that an application could run smoothly while the flow hand offs from one access point to another, if a 'better' interface becomes available. Hop by Hop data transfer ensures, that flow path could change on the fly, without affecting throughput, as illustrated in Fig. 3.1. In case1 (shown in blue), vehicle $\text{GUID}_V$ receives data through NA1. As it drives and moves out of range from NA1 (case 2, shown in red), the chunks on flight, are temporarily stored at $\text{AP}_X$. When $\text{GUID}_V$ comes in range of basestation $\text{B}_Y$, it updates the GNRS with its current point of attachment

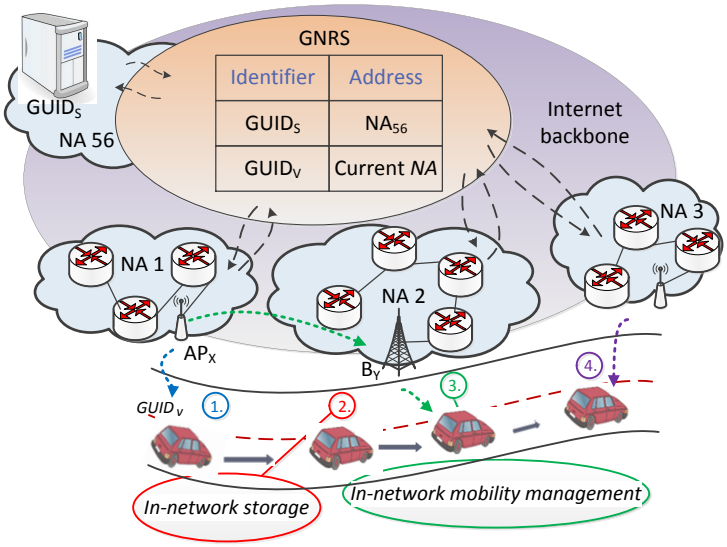



Figure 3.1: Disconnection tolerance and in-network mobility management through MobilityFirst

(NA2 in this case). Chunks from the server, are routed to NA2, while those stored at $AP_X$ get rerouted as well. In Sec. 4.2.1 we show the benefits of using MobilityFirst naming and transport over conventional TCP/IP based data transfer, when using only one interface at a time.

## 3.2 MobilityFirst Routing Mechanism for Multihoming

GNRS lookup binds a chunk to a particular set of network addresses. If all the NAs have a common next hop, a router forwards a chunk downstream, till a bifurcation point is reached. At the bifurcation point, the striping router based on the user-defined policy takes a decision as to which path to send the chunk through. This provides much flexibility to the algorithm,since the bifurcation point could be any inter-mediate router in the internet and could even change during the course of a flow due to end-host mobility. The link quality estimation of the paths, in order to determine at what ratio should the striping take place could be inferred at the junction router through multiple techniques, two which are described below.

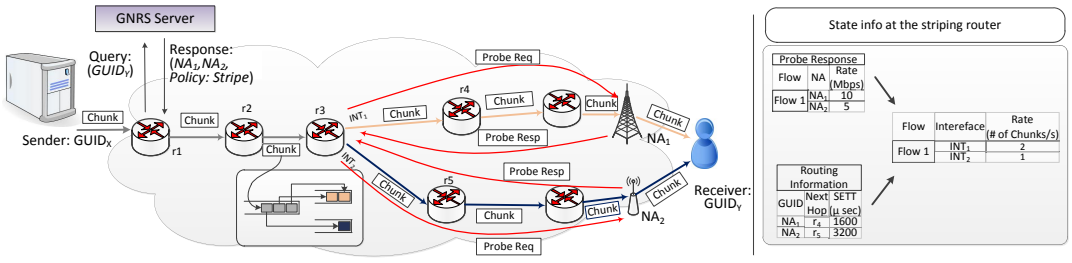



Figure 3.2: Overview of the data rate probing mechanism

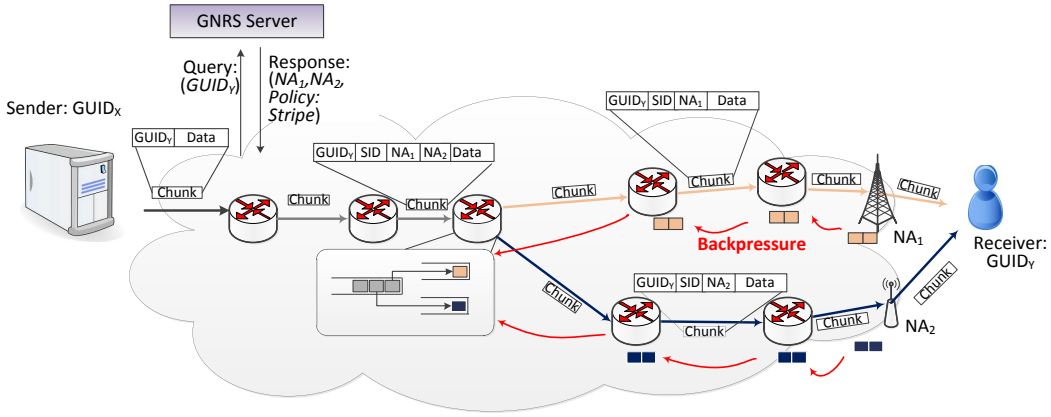



Figure 3.3: Overview of the data-striping technique using backpressure propagation and path quality metrics at each router

## 3.3 The Probe Mechanism

Following our example in Fig. 2.1, if $r_2$ decides to stripe and send chunks to both $r_3$ and $r_4$, it needs to know the actual end-to-end data rate from $r_3$ to $Y$ and $r_4$ to $Y$. $r_2$ leverages the routing information for link quality estimation up to the points of associations and sends a probe packet to both $\text{NA}_1$ and $\text{NA}_2$ to query the current achievable edge data rate that they could support for $Y$. Once, the probe response becomes available, $r_2$, can effectively stripe the flow at the ratio of the calculated maximum end-to-end data rates as shown in Fig. 3.2. The efficacy of such a mechanism would thus be dependent on the *freshness* of the link state information at $r_2$. There could also be a subscription-probe, where the edge reports back data rates automatically, if the supported data rate is highly fluctuating due to mobility or congestion.

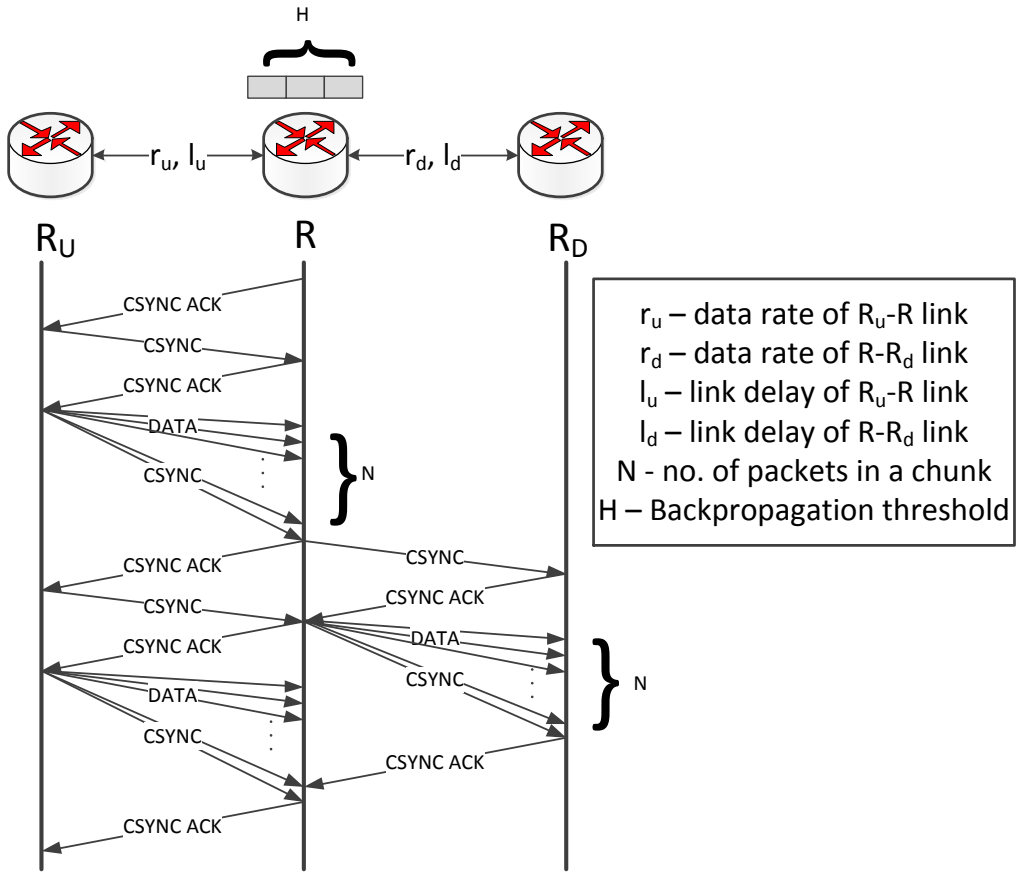



Figure 3.4: Timing diagram of the transmission of chunks

### 3.4 Baseline Algorithm

In contrast to the probe mechanism, we design our algorithm, assuming that no end-to-end path quality is available at $r_2$ or the information is *stale*, due to highly variable link bandwidth-delays. Following our example in Fig. 2.1, if $r_2$ decides to stripe and send chunks to both $r_3$ and $r_4$, it needs to know, the actual end-to-end data rate from $r_3$ to $Y$ and $r_4$ to $Y$. However our algorithm is designed assuming that no accurate end-to-end path quality is available at $r_2$, and it works as follows: $r_2$ starts pushing out chunks as it receives, to both $r_3$ and $r_4$, which in turn transfer them downstream. As explained in Sec. 2.3, every router monitors a per-flow count of the number of pending chunks, and exploits an ack-withholding mechanism, wherein, if it receives at a rate higher than the rate it could push chunks out, and has $H$ pending chunks for that flow, it refuses to

accept chunks, until it could send one more chunk out downstream. This essentially, throttles the flow from the striping router to the end-client across each of the available paths to the rate of the bottleneck of that path, as shown in Fig. 3.3. This considerably simplifies the striping algorithm, as $r_2$ does not require any end-to-end path quality information and instead aims to best utilize the network resources, by pushing chunks out if any/all the next hops accept. The overall flow diagram of the decisions taken at the router is shown in Fig. 3.5. It is important to note that, if the value of $H$ is too low, it might lead to a scenario where a router could be waiting idle for its upstream to send a chunk before it can forward it to its downstream. Going back to our example scenario in Fig. 2.1, say, $NA_2$ could initially support 6Mbps of data rate to $Y$, and due to back-propagation, the striping router had automatically adjusted its sending rate to 6Mbps. Now, if due to client mobility, $NA_2$ could cater to $Y$ at 54Mbps, but it has few pending chunks, $NA_2$ would have to wait idle, before the rate of transmission from the striping router adjusts to this new higher value. In contrast, the larger the value of $H$, the greater the time required for the flow to fill up the pipe from the splitting router to the end-client, which in turn would adversely affect the amount of reordering required.

Theoretically, following the timing diagram from Fig. 3.4, the time taken to transfer a chunk from $R_u$ to R,

$$t_{up} = (\frac{S_{cack}}{r_u} + l_u) + (\frac{S_{csync}}{r_u} + l_u) + (\frac{S_{cack}}{r_u} + l_u) + (\frac{S_{pkt}N + S_{csync}}{r_u} + l_u), \tag{3.1}$$

where,

- $S_{cack}$=size of a csync-ack packet,
- $S_{csync}$=size of a csync packet,
- $S_{pkt}$=size of a data packet

Therefore, maximum time required to transfer a chunk,

$$T_{up_{max}} = (\frac{S_{cack}}{R_{u_{min}}} + L_{u_{max}}) + (\frac{S_{csync}}{R_{u_{min}}} + L_{u_{max}}) + (\frac{S_{cack}}{R_{u_{min}}} + L_{u_{max}}) + (\frac{S_{pkt}N + S_{csync}}{R_{u_{min}}} + L_{u_{max}}) \tag{3.2}$$

where,

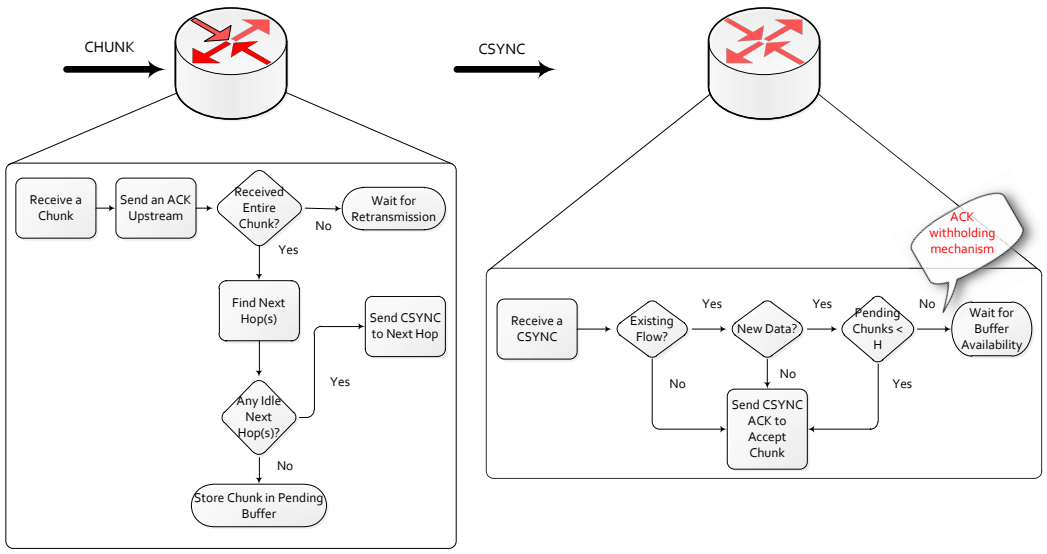



Figure 3.5: Decision logic at each router

- $R_{u_{min}}$ is the minimum possible data rate for the $R$ – $R_u$ link
- $L_{u_{max}}$ is maximum possible link delay for the $R$ – $R_u$ link

On the other hand, during this time, if router $R$ does not wish to sit idle, it should have at least $H$ chunks to send, where time taken to transfer each chunk downstream is,

$$t_d = (\frac{S_{csync}}{r_d} + l_d) + (\frac{S_{cack}}{r_d} + l_d) + (\frac{NS_{pkt} + S_{csync}}{r_d} + l_d) + (\frac{S_{cack}}{r_d} + l_d) \tag{3.3}$$

and similarly, the minimum time required to transfer a chunk downstream is,

$$T_{d_{min}} = (\frac{S_{csync}}{R_{d_{max}}} + L_{d_{min}}) + (\frac{S_{cack}}{R_{d_{max}}} + L_{d_{min}}) + (\frac{NS_{pkt} + S_{csync}}{R_{d_{max}}} + L_{d_{min}}) + (\frac{S_{cack}}{R_{d_{max}}} + L_{d_{min}}) \tag{3.4}$$

Therefore, the theoretical minimum value of H required is,

$$H_{min} = \lceil \frac{T_{up_{max}}}{T_{d_{min}}} \rceil \tag{3.5}$$

In Chapter 4, we study the effect of $H$ on reordering and throughput and present simulation results justifying the value of $H$ chosen for the simulations.

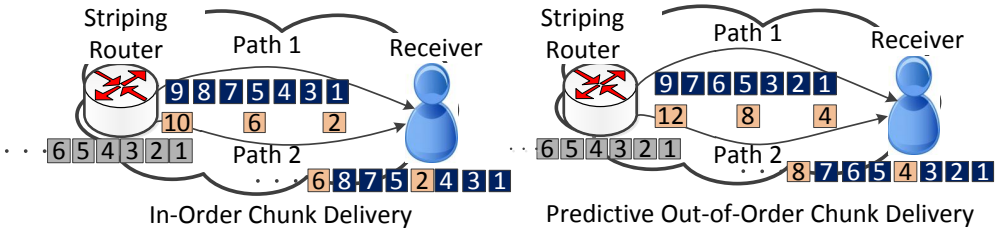



Figure 3.6: Comparison of in-order vs. out-of-order chunk delivery

## 3.5 Modification to Reduce Reordering

The algorithm to simply send data if the next hop accepts, would enable effective utilization of link capacities in terms of the raw throughput. However, if the link bandwidths and delays are not comparable, it would lead to a large number of out-of-order chunks at the receiver, as demonstrated later in Chapter 4. If the application demands in-order delivery of data, the overall application throughput would essentially be dragged down by the slower link, and in the worst case, if the receiver side has buffer constraints, buffered packets would eventually be dropped. Thus, one of the key requirements of the striping algorithm is that, the network should be able to gracefully adapt to bandwidth delay variations across various links and at the same time, reduce the amount of reordering at the receiver.

We propose a modification of our baseline multihoming algorithm, with the intuition that, if the striping router can estimate the amount of reordering the receiver is facing, it can send out-of-order chunks across the different paths to reduce this effect. The striping router monitors the number of chunks requested by each interface for a flow to estimate the data rate ratios of the two paths, and intuitively sends chunks from the back of the queue across the slower interface. The ratio of chunks sent on the two paths is set in a way so as to minimize reordering requirements at the client. Fig 3.6 illustrates an example, where, the striping router estimates path 1 to be three times as fast as path 2, and sends chunks from the back of the queue through the slower path. Notice, that instead of sending the first chunk in line, it sends every 3rd chunk from the front of the queue, across path 2, as it estimates path 2 to be three times as slow as path

1. There have been other works on multipath routing [31] where a similar approach is employed, with the assumption that multiple paths are available end-to-end from the sender to the receiver and the sender has some prior knowledge of the link qualities. Our algorithm, on the other hand, assumes that the in-network routers transmit data hop-by-hop and as such, have no priori estimate of the links. It starts its striping with equal weightage or in-order delivery of chunks, and switches to out-of-order, once the observed outgoing rates of the interfaces start to differ. This is based on the intuition that, chunks sent across the slower end-to-end path would arrive later.

# Chapter 4

# Evaluation

In this chapter we evaluate the performance of the in-network approach to multihoming explained in Chapters 2 and 3 through detailed NS3 simulations. The basic simulation setup is explained first, followed by throughput evaluation, parametric exploration and evaluation of a realistic heterogeneous scenario.

## 4.1 Simulation Setup

For the purpose of simulation, we model MobilityFirst's naming, routing, and transport mechanisms in NS3, and use NS3's existing comprehensive Wi-Fi and LTE modules [32] to realistically capture the properties of the respective wireless interfaces. Table 4.1 shows the values of the key parameters of the model that were kept constant throughout the evaluation.

| **Radio** | **Function** | **Value** |
|---|---|---|
| WiFi | MAC | 802.11a(nonQoS) |
| | Wireless Inteface Mode | Infrastructure |
| | Propagation Models | Log Distance Loss |
| | | Constant Speed Delay |
| | Rate Control Algorithm | Adaptive ARF |
| LTE | DL and UL Resource Blocks | 15 |
| | Mac Scheduler | Proportional Fair |
| | DL Tx Power | 30dBm |
| | UL Tx Power | 10dBm |

Table 4.1: Simulation Model Details

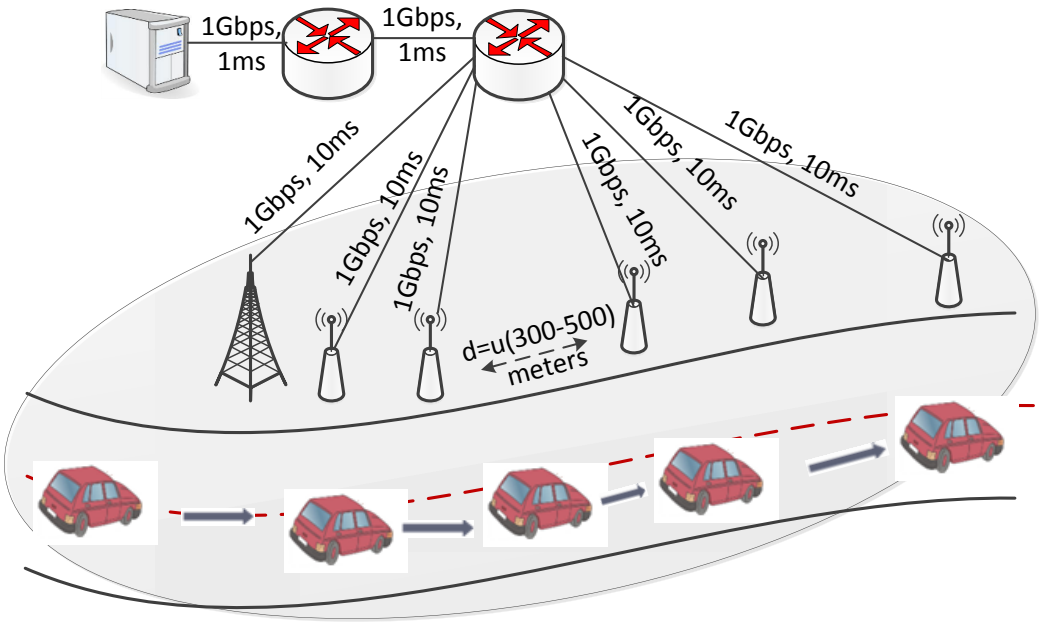



Figure 4.1: Simulation topology of a mobile client with a Wi-Fi and an LTE interface

## 4.2 Throughput Gains Through Use of Multiple Interfaces

In this section, we study the gains in terms of throughput that could be achieved when allowing the use of multiple interfaces, for best interface data transfer in Sec. 4.2.1 and using all the available interfaces simultaneously for striping in Sec. 4.2.2. In the first case, we compare the performance of our interface selection and transport mechanisms with a TCP/IP implementation. However, for the second case, since simultaneous transfer over two interfaces, i.e. interleaving of packets is not supported in baseline TCP, we compare the gains against the maximum achievable throughput, as explained later.

### 4.2.1 Opportunistic Wi-Fi and its Comparison with TCP

Here we compare the performance of the MobilityFirst architecture with the current TCP/IP based Internet access by vehicular nodes when they opportunistically use Wi-Fi hotspots. Fig. 4.1 shows the simulation topology in which a multihomed mobile client moves along a straight road at varying speeds, with access points deployed along the road at random inter-AP distances $d_i$. Values of $d_i$ are chosen from a uniform random distribution such that during the simulation the vehicular node experiences good connectivity, poor connectivity as well as temporary disconnections through Wi-Fi (depending on the parameters of the uniform random variable, as explained later). We assume the client's policy is to use only Wi-Fi, as and when it becomes available, and hence no backend LTE is simulated in this scenario. The number of hops from the server to the client is kept at 4, however, the link delay from the core to the edge is amortized to 10ms to have a realistic network model. For the TCP/IP comparison, we do not assume a managed MobileIP implementation since the setup being evaluated here is that of opportunistic connections through open 802.11 APs which are deployed and managed by different entities. As in practice, the moving client in our simulation gets a new IP address via DHCP upon connecting to each AP in the TCP/IP case. The latency for IP address acquisition is chosen from a uniform random distribution between 1 to 2 seconds (Authors in [33] experimentally demonstrate that the DHCP latency for typical vehicular scenarios is 1.8 seconds on an average). Further, in order to make a fair comparison, we assume a 'smart' application layer over TCP/IP that resumes transmissions from its point of last connection instead of re-requesting entire transfers. Correspondingly, for the MobilityFirst case, every time the client switches AP, GNRS update/query events are generated which take between 30-170 ms as per the evaluation in [27].

The data transmission model emulates a web browsing scenario with intermittent bursts of small transfers from server to the client. The client sequentially requests different sized content packets from the server and we measure the time difference between when each request was made and served, for both MobilityFirst and TCP/IP.

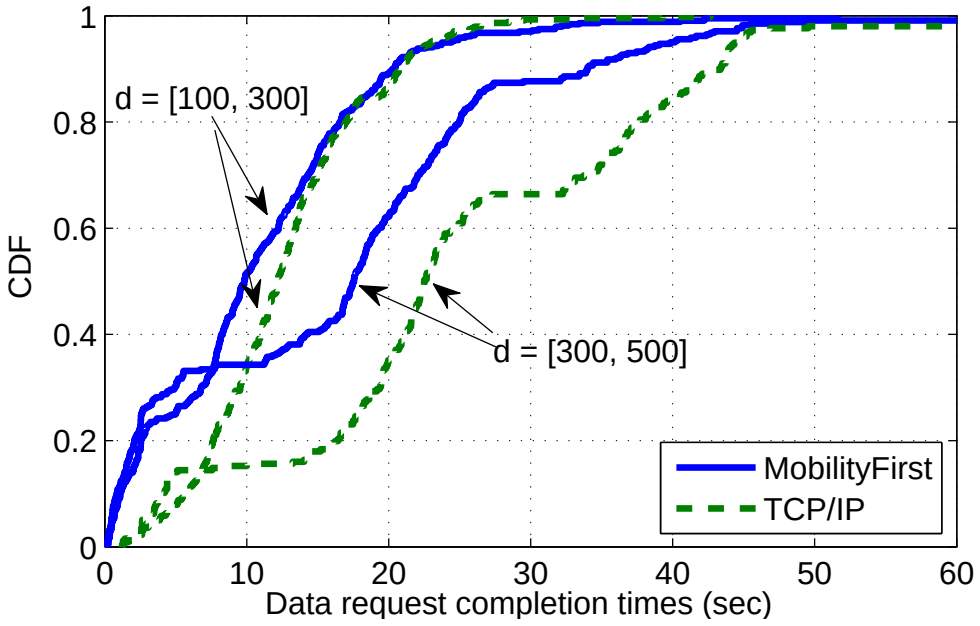



Figure 4.2: Cumulative distribution of the data request completion times in the web-browsing emulation scenario

The size of the requested content is uniformly distributed between 10KB to 5MB, roughly representing HTTP packet lengths in use today [34]. The speed of the car is kept at a constant 50 miles/hr, while two settings of inter-AP distance $d$ is used: uniformly distributed between 100-300m and same with 300-500m. The cumulative distribution function of the request completion times are shown in Fig. 4.2.

The results show a significant reduction in the transfer times for both values of $d$ - median gains of around 4 seconds or 30% in the [100, 300] case and 5 seconds or 22% in the [300, 500] case. Another way to interpret these results are to look at the percentage of completed requests within a given time-frame. On this scale, there is almost a 2x gain in both cases, for example, when measuring the fraction completed at 5 seconds. This gain can be traced to two key differences between MobilityFirst and TCP/IP. First, when the node disconnects from an AP, the packets destined for it are stored locally at the last AP instead of being dropped, and are sent quickly to the next AP when the node connects there. And second, the amount of 'useful time' spent in each AP is increased since the node retains its GUID as it traverses multiple APs.

### 4.2.2 Striped Data Transfer Through Wi-Fi and LTE

Next, we study the raw throughput gains that can be achieved by vehicular nodes when multiple interfaces are used simultaneously, wherein they opportunistically use Wi-Fi hotspots while already being connected to an LTE network. The simulation topology remains the same as shown in Fig. 4.1. Values of $d$ are chosen from a uniform distribution between 300-500 meters. The LTE connection is simulated to have a stable coverage but with lower achievable data rate, as is prevalent in typical vehicular scenarios. Fig. 4.3 shows the aggregate throughput at the client, when it receives a continuous stream of data for the entire simulation, moving at 10meters/sec( 22mph), while Fig. 4.4 plots the transfer completion times when the client requests a single file of random size between 60-80 MBytes averaged over 10 random runs. As the figures indicate, the in-network data-striping algorithm fully utilizes the Wi-Fi interface whenever it becomes available. This is indicated in Fig. 4.3 by the multihoming throughput being close to the sum of the throughput achievable through each of the individual interfaces. Note that the green curve plotted is not a simulation result but the sum of the red and black curves and is for the purpose of illustration of what total throughput can be achieved if the interfaces were used simultaneously but for different applications with no correlation as such. The gap between the green and the blue curves is because of the way the GNRS operates in our simulation scenario. The GNRS updates are initiated by the client during 'making a connection' and not during 'breaking a connection'. Also, in-network routers initiate GNRS queries, if they detect a packet loss. When the vehicle is at the edge of the Wi-Fi coverage area, thus any packet loss at the access point results in a GNRS query. For the multihomed scenario, the GNRS response prompts the AP to reroute the pending chunks to the LTE base station. However, for the Wi-Fi only scenario, since there is no-other available interface, the AP keeps re-sending packets across the Wi-Fi channel and manages to transfer some more data, before the vehicle moves completely out-of-range. Since the Wi-Fi channel (even at the edge of the coverage area), is inherently faster than the LTE medium, and re-routing of chunks incur additional delays, the mathematical sum of aggregate throughputs appear to be slightly better than the multihomed scenario. However, since no packets are ultimately

lost in any scenario (due to reliable hop-by-hop transfer with temporary caching), the blue curve tends to eventually catch up with the green, as seen in the plots in the 30-50 seconds zone, when the disconnection from the first access point occurs.

Fig. 4.4 on the other hand, shows the gains in terms of latency of file downloads when using all the available interfaces. The vehicle on detecting Wi-Fi coverage waits for a random delay (uniform random variable between 0 and 5 seconds) initiates a file download of uniform random size between 60 to 80 Mbytes. This results in large error bars for the Wi-Fi only and the multi-homed scenario. If the request is small, and is made when the Wi-Fi connectivity is good, the entire file could be downloaded within one association instant. On the other hand, in the worst case, a node might require several Wi-Fi association instances for a download to complete and since the Wi-Fi coverage is intermittent, this would translate to longer file transfer completion times. We do note that, allowing file downloads to be initiated only during WI-Fi availability biases the experiment towards the Wi-Fi only scenario, as there are regions of no Wi-Fi coverage where a vehicle could have initiated and failed in downloading any data, and those data points if allowed, would have adversely affected the Wi-Fi only file transfer times.

## 4.3 Parametric Evaluation

In this section, we perform several sets of micro-simulations to examine the parameter regime and the choice of parameters on the performance.

### 4.3.1 Effect of Link Parameters on Reordering

The following set of simulations study the effect of reordering on the application buffer requirements at the client, due to data striping across multiple disparate paths. Reference [35] suggests several metrics to measure the extent of packet reordering , based on which we choose two methods of expressing reordering-the average buffer occupancy and the reorder density, each of which is described below:

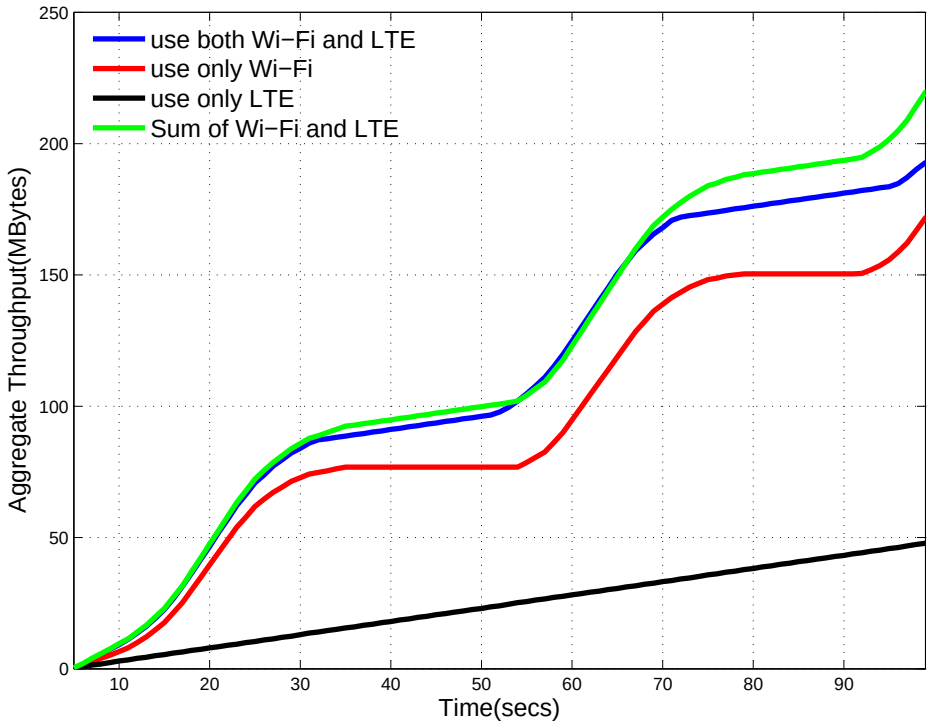



Figure 4.3: Aggregate throughput for a multihomed mobile client with a WiFi and an LTE interface

- Average buffer occupancy: Consider the toy example depicted in Table 4.3. The average buffer occupancy at the client for the 10 chunk window is $(0+0+1+0+2+0+1+1+0)/10 = 0.5$. For a chunk size of 1.6Mbytes (as considered for our simulations), this would amount to an average buffer occupancy of 0.8Mbytes, with the worst case requirements of 3.2Mbytes.

- Reorder Density: Considering the same sequence of arrival of chunks, the reorder density is obtained by calculating the displacement from the actual sequence as

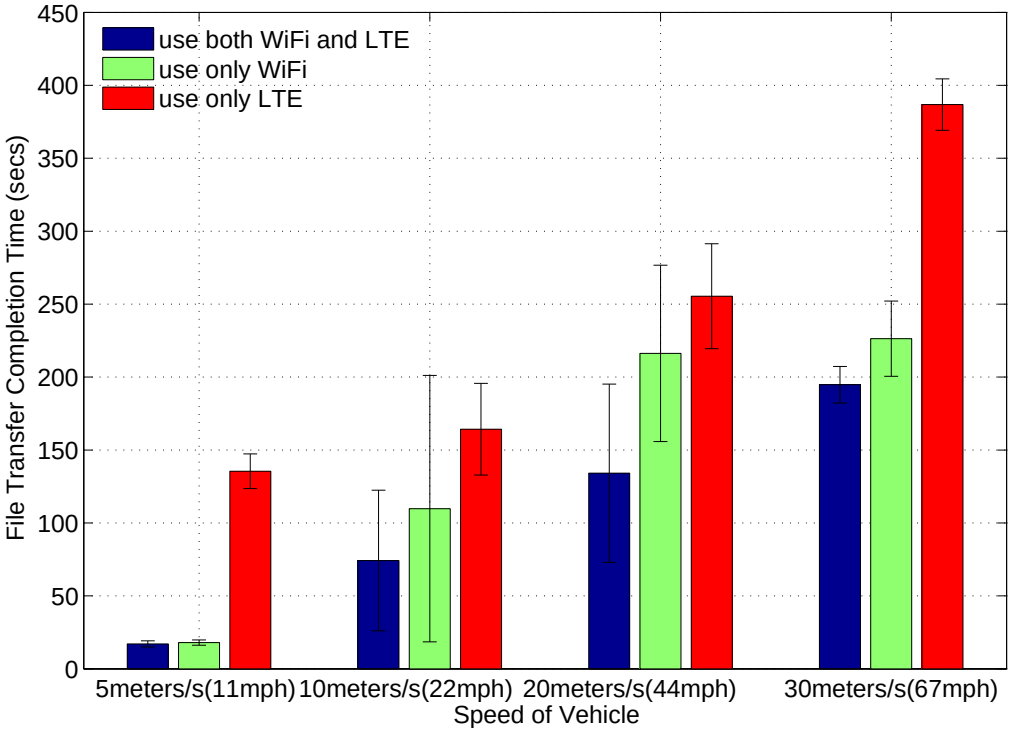



Figure 4.4: File transfer completion for mobile client with a WiFi and an LTE interface

shown in Table 4.4.

$$\text{Reorder density for -2 out-of-order} = 1/10 = 10\%$$

$$\text{Reorder density for -1 out-of-order} = 3/10 = 30\%$$

$$\text{Reorder density for 0 out-of-order} = 2/10 = 20\%$$

$$\text{Reorder density for +1 out-of-order} = 3/10 = 30\%$$

$$\text{Reorder density for +2 out-of-order} = 1/10 = 10\%$$

We study three key parameters which could affect reordering, namely, link data rate, link latency and number of hops from the striping router to the end host. The basic topology remains the same, but this time the client is equipped with two WiFi interfaces and is kept static. The baseline simulation parameters are shown in Table 4.2. Each parameter is then changed, such that they are in ratios of 1:1, 1:2 to 1:5, keeping the others constant and the buffer size at the application is measured, from which the

| Simulation Parameters | Value |
|---|---|
| No. of interfaces | 2 WiFi |
| Client mobility | NIL |
| No. of hops from striping router to client | 2 |
| WiFi edge data rate | 36Mbps |
| Core links | Ethernet, 1Gbps, 10ms |
| Chunk size | 1.6Mbits |
| Backpropagation threshold | 2 chunks |

Table 4.2: Simulation parameters for baseline scenario

| **In-order sequence of arrival** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Actual sequence of arrival** | 1 | 2 | 4 | 3 | 6 | 7 | 5 | 10 | 8 | 9 |
| **Buffer size at the application** | 0 | 0 | 1 | 0 | 1 | 2 | 0 | 1 | 1 | 0 |

Table 4.3: Example of application buffer size for a given sequence of arrival of chunks

average buffer occupancy is calculated. As seen in Fig. 4.5, the dominant factor in reordering is the disparity in the data rate of the edge links. Note that the increase in reordering due to increase in number of hops from the point of striping to the end client, is due to the fact, that the striping router has no estimation of the link qualities, until the pipes along the striping paths are full, and the *ack-withholding* congestion control comes into play making the sending rate at the router's interface essentially equal to the output rate at the client's interfaces. However, even with a hop ratio = 1:5, it leads to marginal reordering requirements of less than 1 MB for the simulation topology considered. As mentioned earlier, the increase of out-of-order arrival of packets at the client, motivated us to modify our baseline algorithm to deliver out of order chunks from the striping router in an attempt to reduce application layer buffer requirements.

Fig. 4.6 shows the improvements in terms of reduction of buffer requirements, when we employ predictive out-of-order striping of chunks. As explained, the optimization would not be able to counter the reordering due to number of hops, however, it brings down the reordering due to both link delay and data rate disparities. Fig. 4.7 shows the reorder densities for data rate disparities and its improvements with out-of-order striping. This plot highlights an important feature of the algorithm. If the striping router sends in-order chunks, most of the chunks have very less displacements, however,

| **In-order sequence of arrival** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Actual sequence of arrival** | 1 | 2 | 4 | 3 | 6 | 7 | 5 | 10 | 8 | 9 |
| **Displacement** | 0 | 0 | -1 | +1 | -1 | -1 | +2 | -2 | +1 | +1 |

Table 4.4: Example of displacement in sequence for a given sequence of arrival of chunks

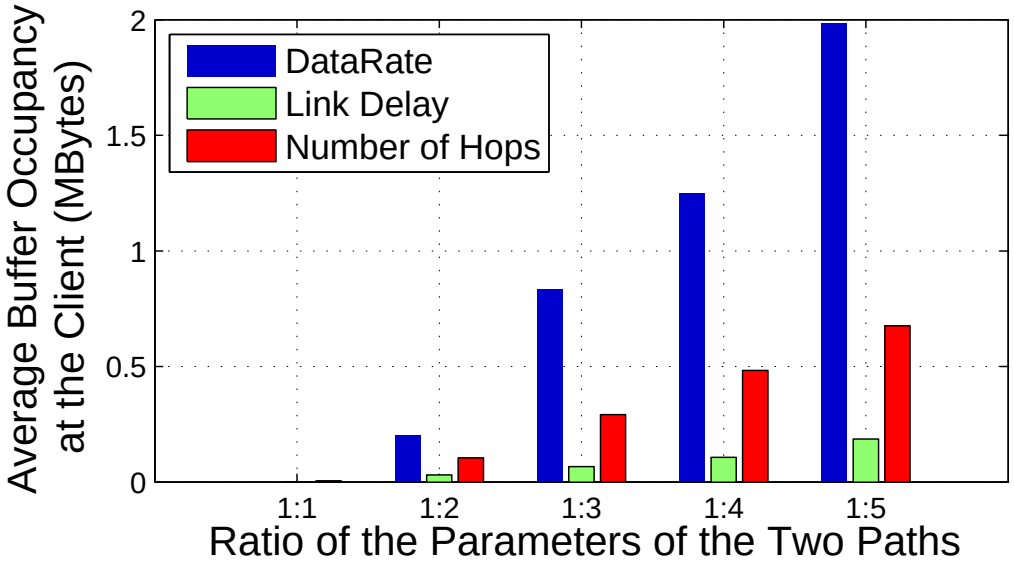



Figure 4.5: Effect of link bandwidth, delay and number of hops on the reorder buffer requirements at the client

some chunks arrive really late, as witnessed by the horizontal spread in the positive direction of the x-axis. On the other hand, with out-of-order striping, some chunks arrive ahead of time (as indicated by the negative spread in the plots), but, the overall spread is reduced, indicating no chunk has to wait too long in the buffer, before being passed onto the application. It is to be noted, that the out-of-order delivery would not in any way reduce the raw throughput at the client, as the striping router only sends chunks out-of-order, if available, and does not wait idle for chunks to arrive at the back of the queue, if it already has pending chunks to send.

### Video Streaming Application with Out-of-Order Chunk Delivery

Having highlighted the importance of optimized out of order data delivery, we study the performance gains and trade-offs when using two interfaces and trying to stream a video, keeping in mind that for video streaming purposes, it is essential that the application receives in order data with a minimum lag. The simulation scenario is similar to the one

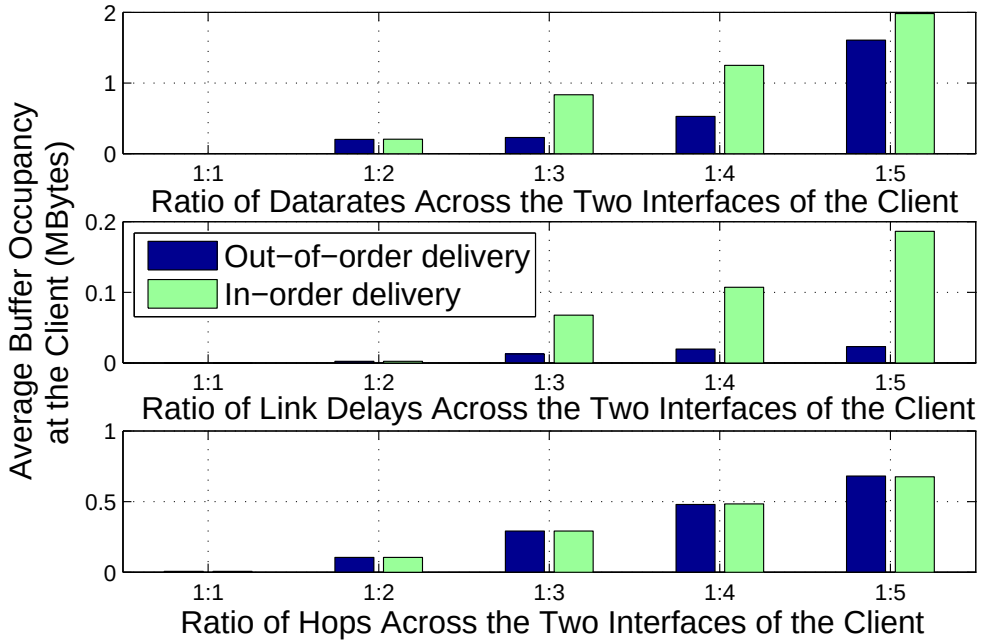



Figure 4.6: Reduction in buffer requirements with out-of-order delivery of chunks

described in Sec. 4.2.2. Fig. 4.8 shows the aggregated raw throughput in comparison to in-order application throughput when using opportunistic Wi-Fi with LTE connections simultaneously for a vehicular node moving at 20m/s( 44mph) and streaming a video of size 100Mbytes. As seen from the plot, the application throughput advances in small jumps, as some chunks arrive out-of-order, and need to wait in the buffer for previous chunks (chunks with lower chunk id) to arrive through the other interface. However, the in-order application throughput closely follows the raw throughput trend (which in itself is close to the sum of the raw throughput if the client used each of its interfaces independently, as shown before in Fig. 4.3).

### 4.3.2 Effect of Backpropagation Threshold on Performance

Next we investigate the effect of the back-propagation threshold on performance (referred to as $H$ in Sec. 2.3). The simulation scenario consists of a single static client with two WiFi interfaces having physical data rates of 54Mbps and 6Mbps respectively, requesting a 200MB file from the server. In Fig. 4.9, we see that the file transfer completion time is maximum for $H$=1, since when $H$ is set to 1, a router allows a maximum of

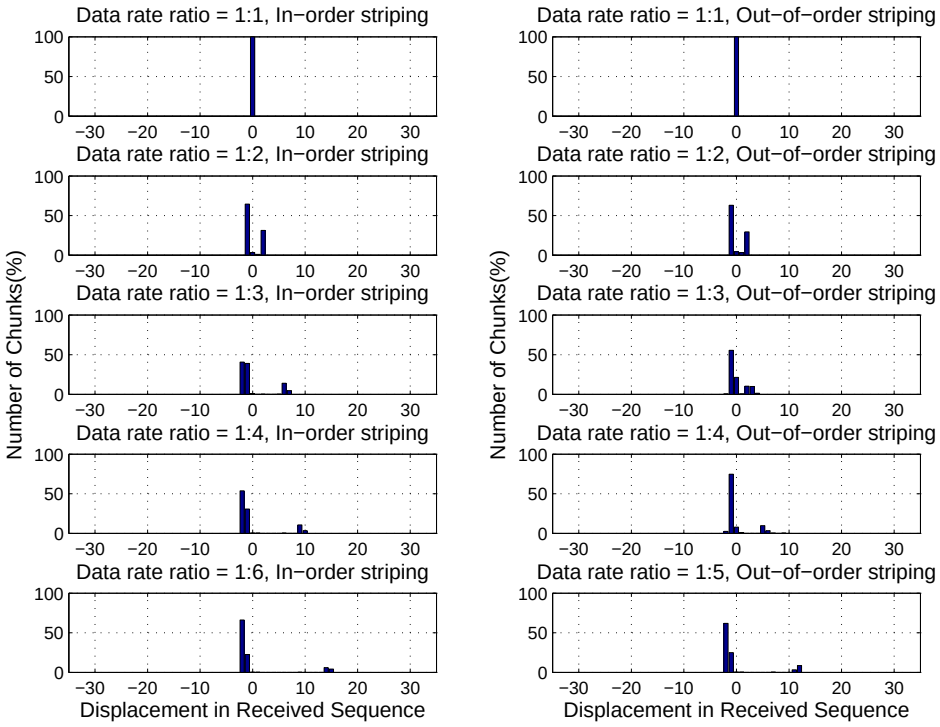



Figure 4.7: Reorder density for different data rate ratios, with in-order and predictive out-of-order striping

only one chunk in its pending buffer at any time. Concretely, this means that it starts accepting a chunk, only when it has been able to successfully transfer the previous chunk downstream and thus waits idle until it receives the next chunk from upstream. On the other hand, the higher the value of $H$, the larger is the time of completion, as the striping router keeps sending chunks equally across both the interfaces (instead of the ideal rate of 6:54) for a longer time, until the ack-withholding comes into play, as shown in Fig. 4.9. At the same time, the lower the value of $H$, lesser would be the reordering at the application, as the striping router would be able interpret the data rates faster, with the minimum being at $H$=1. For the chosen chunk size and delay bandwidth products of the upstream and downstream links at each router, the value of 2 appears to be the minimum. However there could be certain cases where even for $H$=2, a router would be waiting idle and hence 2 might not always be the best choice.

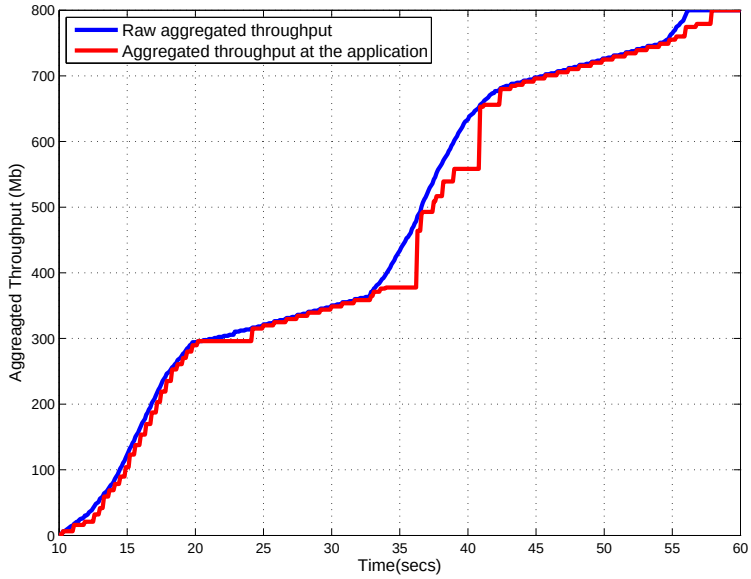



Figure 4.8: Comparison of raw throughput with inorder application throughput

### 4.3.3 Use of Both vs. Best Interface

Finally, we examine the parameter regime (i.e. bit-rate ratio, packet size, etc.) for which striping across multiple interfaces is beneficial. Fig. 4.10 summarizes the comparison of sending through the best vs. sending through both interfaces for the client for different file download sizes and different data rate ratios of the edge Wi-Fi links. The z-axis plots the ratio of file transfer completion times of sending through the best interface to using both the available Wi-Fi interfaces at the client. The gray plane cutting across the plot is at z=1. Ratios higher than 1 indicates there is a definitive advantage in striping, as it results in lower completion times. From the plot, we can draw two interesting observations: Firstly, if the request is small, the application actually suffers a performance degradation due to striping and secondly, lower the disparity of the data rates between the available interfaces, higher the gains. This motivates us to rethink the initial goals of achieving multihoming and indicates the need for having soft thresholds for the flow size and the data rate disparities below/above which the routers would decide to use the best interface instead of both, even though the application might be willing to receive from both.

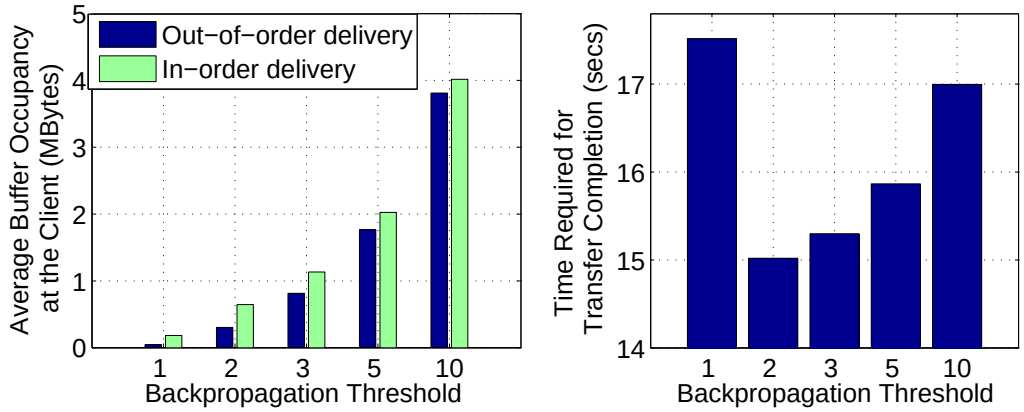



Figure 4.9: Effect of variation of H on throughput and reordering at the application

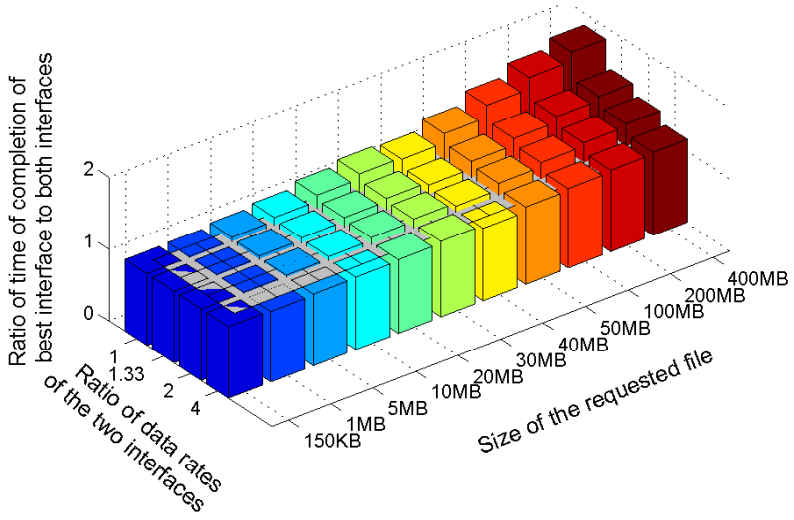



Figure 4.10: Comparison of using the best vs. both interfaces for different data flow sizes and ratios of data rates of the interfaces

## 4.4 Real World Model

Finally we demonstrate the benefits of utilizing multiple interfaces in a realistic hetnet scenario, shown in Fig. 4.11. The figure shows an approximate 1x2 sq. miles area near the financial district of San Francisco, with the red dots, showing the open Wi-Fi hot spots in the area. The data of the AP locations were obtained from the AT & T Wi-Fi locator [36] and include open Wi-Fi hot spots mostly at road-side coffee-shops, restaurants, etc. Since exact location of base stations were not available, we assumed an LTE base station to be located at the center of the area, and to which all the mobile nodes could connect to. To realistically analyze vehicular mobility, we use GPS traces

of taxi-cabs in the area from the publicly available data set at Community Resource for Archiving Wireless Data At Dartmouth(`crawdad.org`) [37]. The data set includes GPS traces ( in latitude and longitude with time stamp) of around 500 taxis collected for over a period of 30 days from 2008/05/17 to 2008/06/10. These traces were transformed to utm coordinates, along with the AP locations and plugged into the simulator. Fig. 4.12 summarizes the instantaneous throughput experienced by 5 randomly chosen vehicular nodes (the cab numbers on the x-axis are for illustration purposes) in the area chosen, for a continuous data transfer of 100 seconds. The back-end connection of the APs and the base station with the data server is kept the same as before. The throughput of each node is measured every second, the average of which is shown on the left, while the maximum is plotted on the right. This plot provides some interesting insights on a practical I2V or V2I interactive application performance. Firstly, if a vehicle's policy is to use only Wi-Fi at all times and is equipped with only one 802.11 radio, depending on the AP locations and its mobility pattern, it might receive no connectivity at all for a majority of the time ( see cab no.1). Secondly, even if the vehicle has multiple Wi-Fi cards, striping using multiple Wi-Fis might not yield significant benefits,unless the location considered has a lot of open Wi-Fi access points with overlapping coverage. From the plots, we see only cab no.2 could achieve some aggregated bandwidth benefits by striping through multiple Wi-Fis. Finally, if a vehicular node is willing to use multiple Wi-Fi interfaces simultaneously with an LTE interface, it achieves significant throughput gains both for the maximum achievable speed as well as average download speed, independent of the mobility pattern of the vehicle.

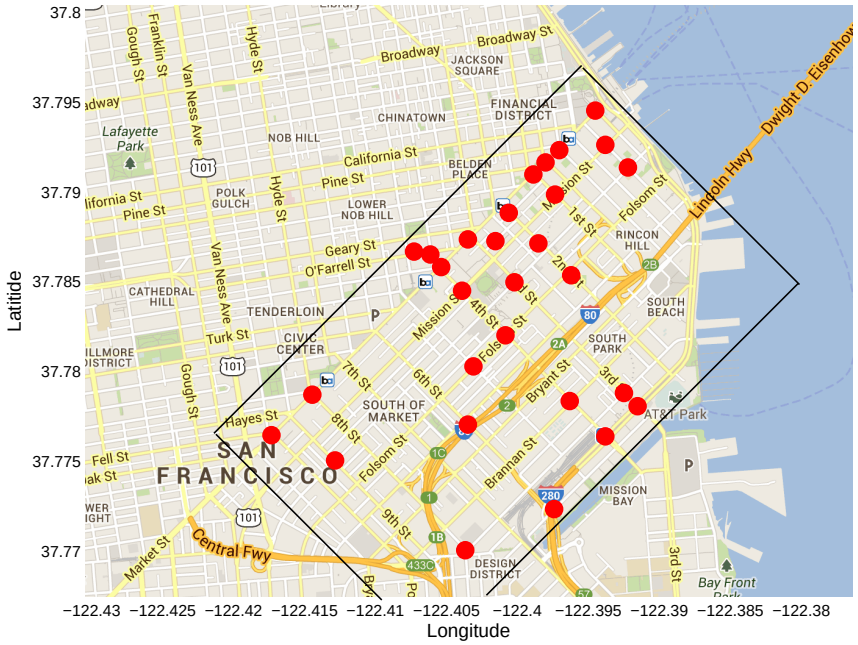



Figure 4.11: San-Francisco financial district with open Wi-Fi hotspots in red

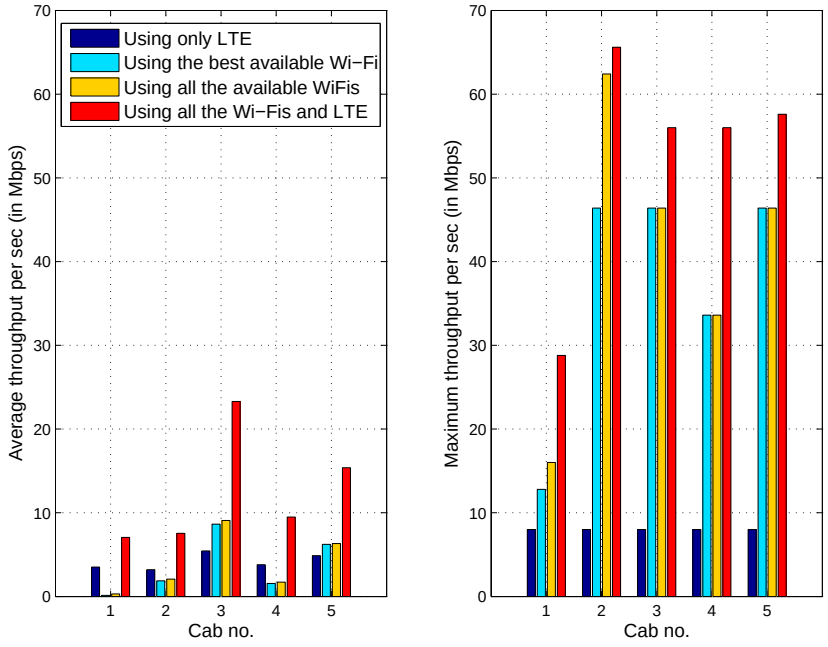



Figure 4.12: Average and maximum throughput experienced per second (in Mbps) by 5 random cabs

# Chapter 5

# Future Work and Conclusion

## 5.1 Future Work

Certain aspects of this work still need to be further explored.

- **Parameter regime:** The effects of hop-count delay and data rates at the edge links, although being studied, the results provided were for static clients. Further tests need to be done to determine the effects of these parameters on mobile clients with dynamically changing wireless scenarios, along with other parameters that might affect the performance.
- **Single-to-multi interface threshold:** As illustrated in evaluation section, there is a threshold depending on the data rate ratio of the links and the flow size below which striping across two interfaces does not yield additional benefits. This needs to be generalized for multi-interface scenarios and further study needs to be done to analyze, how the routers would determine this threshold on the fly.
- **Scalability:** There needs to be a study on the scalability and robustness of the algorithm for internet-scale network topologies.
- **Comparison with existing algorithms:** This work also needs a comparison of the algorithm with existing methods of data striping for end-to-end flows.

## 5.2 Conclusion

In this thesis, we investigated a flexible network-assisted multihoming scheme, utilizing the protocol features associated with a name based network architecture with hop by hop reliable data transfer and storage aware routing. We described a specific data

striping algorithm which allows simultaneous data transfer across multiple interfaces with per-flow based back-pressure link quality estimation. NS3 simulation results for realistic urban mobility models with heterogeneous Wi-Fi/LTE coverage demonstrated aggregate throughput benefits with effective reduction in reorder-buffer requirements at the application. Finally, our results also indicated that utilizing both the available interfaces might not always lead to a performance improvement, particularly when the bit-rate ratio is too high (>1:4) or the flow size is too low (<5Mbytes) for the parameters considered.

# References


[1] "Mobilityfirst future internet architecture project," http://mobilityfirst.winlab.rutgers.edu/.

[2] R. Moskowitz and P. Nikander, *Host Identity Protocol (HIP) Architecture*, IETF Internet Standard, RFC 4423, May 2006.

[3] T. You and H. Jung, "Exploiting mobile oriented future internet (mofi) for future cloud datacenter networks," in *Proceedings of the 7th International Conference on Future Internet Technologies*, ser. CFI '12, 2012.

[4] T. Ernst, N. Montavont, R. Wakikawa, C.-W. Ng, and K. Kuladinithi, "Motivations and scenarios for using multiple interfaces and global addresses," 2008.

[5] Chiussi,F.M., Khotimsky D.A. and Krishnan S., "Generalized Inverse Multiplexing of Switched ATM Connections," in *Proceedings of the IEEE Conference on Global Communications*, ser. GlobeCom '98, 1998, pp. 3134 – 3140 vol.5.

[6] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradett, *The PPP Multilink Protocol(MP)*, IETF Internet Standard, RFC 1717, August 1996.

[7] Pathak,D.S. and Goff,T., "A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, ser. Infocom '02, 2002, pp. 773 – 781 vol.2.

[8] Chebrolu,K., Raman,B. and Rao,R., "A Network Layer Approach to Enable TCP over Multiple Interfaces," *Wireless Networks*, vol. 11, pp. 637–650, 2005.

[9] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi, "Fatvap: Aggregating ap backhaul capacity to maximize throughput." in *NSDI*, vol. 8, 2008, pp. 89–104.

[10] R. Chandra and P. Bahl, "Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, 2004, pp. 882–893.

[11] A. Qureshi and J. Guttag, "Horde: separating network striping policy from mechanism," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 121–134.

[12] A. Qureshi, J. Carlisle, and J. Guttag, "Tavarua: video streaming with wwan striping," in *Proceedings of the 14th annual ACM international conference on Multimedia*. ACM, 2006, pp. 327–336.

[13] M. Py, *Multi Homing Translation Protocol (MHTP)*, IETF Internet Draft, draft-py-multi6-mhtp-01.txt, May 2002.

[14] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 5, pp. 951–964, 2006.

[15] Hsieh,H. and Sivakumar,R., "pTCP: an end-to-end transport layer protocol for striped connections," in *Proceedings of 10th IEEE International Conference on Network Protocols.*, 2002, pp. 24 – 33.

[16] Magalhaes,L. and Kravets,R., "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proceedings of Ninth International Conference on Network Protocols*, 2001, pp. 165 – 171.

[17] Ahmed,A., Saadawi,T. and Lee,M., "LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol," *Protocol Engineering for Wired and Wireless Networks*, vol. 27, p. 10121024, 2004.

[18] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, *Architectural guidelines for multipath TCP development*, IETF Internet Standard, RFC 6182, March 2011.

[19] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee, "Mar: A commuter router infrastructure for the mobile internet," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services.* ACM, 2004, pp. 217–230.

[20] Z. Fu, P. Zerfos, H. Luo, and S. Lu, "The impact of multihop wireless channel on tcp throughput and loss," in *Proceedings of the IEEE INFOCOM*, 2003, pp. 1744–1753.

[21] Alman,M., Kruse,H. and Ostermann,S., "An Application-Level Solution to TCP's Satellite Inefficiencies," in *Proceedings of the First International Conference on Satellite-Based Information Services*, ser. WOSBIS '96, 1996.

[22] Thompson,N., He,G. and Luo,H., "Flow Scheduling for End-Host Multihoming," in *Proceedings of the IEEE International Conference on Computer Communications*, ser. INFOCOM '06, 2006.

[23] Cisco White Paper, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016," Feb. 2012.

[24] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "GSTAR: Generalized storage-aware routing for MobilityFirst in the future mobile Internet," in *Proceedings of MobiArch'11*, 2011, pp. 19–24.

[25] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri, "MobilityFirst Future Internet Architecture Project," in *MobilityFirst Project, Proc. ACM AINTec 2011.*

[26] N. Somani, A. Chanda, S. C. Nelson, and D. Raychaudhuri, "Storage-Aware Routing for Robust and Efficient Services in the Future Mobile Internet," in *Proceedings of ICC FutureNet V, 2012.*

[27] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on.* IEEE, 2012, pp. 698–707.

[28] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable internet protocol (aip)," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 339–350.

[29] T. Vu, A. Baid, H. Nguyen, and D. Raychaudhuri, "Eir: Edge-aware interdomain routing protocol for the future mobile internet," WINLAB, Tech. Rep. WINLAB-TR-414, June 2013.

[30] M. Li, D. Agarwal, and V. A., "Block-switched networks: A new paradigm for wireless transport."

[31] Wang,J., Liao,J. and Li,T., "OSIA: Out-of-order Scheduling for In-order Arriving in concurrent multi-path transfer," *Journal of Network and Computer Applications*, vol. 35, p. 633643, 2011.

[32] G. Piro, N. Baldo, and M. Miozzo, "An lte module for the ns-3 network simulator," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '11, 2011, pp. 415–422.

[33] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular internet access using in situ wi-fi networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking.* ACM, 2006, pp. 50–61.

[34] N. Basher et al., "A comparative analysis of web and peer-to-peer traffic," in *In Proceedings of the 2008 WWW Conference*, 2008.

[35] N. M. Piratla and A. P. Jayasumana, "Metrics for packet reorderinga comparative analysis," *International Journal of Communication Systems*, vol. 21, no. 1, pp. 99–113, 2008.

[36] "At&t wi-fi locator," http://www.att.com/maps/wifi.html.

[37] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/mobility (v. 2009-02-24)," Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, feb 2009.