PERFORMANCE EVALUATION OF FORWARDING ALGORITHMS FOR GENERALIZED STORAGE AWARE ROUTING PROTOCOLS

BY NEHAL SOMANI

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

May, 2012

ABSTRACT OF THE THESIS

Performance Evaluation of Forwarding Algorithms for Generalized Storage Aware Routing Protocols

by Nehal Somani

Thesis Director: Professor Dipankar Raychaudhuri

This thesis presents an investigation of the design and evaluation of the generalized storage aware routing (GSTAR) protocol proposed for use in the MobilityFirst future Internet architecture. The GSTAR protocol uses in-network storage to improve service quality and throughput in wireless access networks with varying radio link quality and/or disconnection. These gains are achieved using a combination of short-term buffering at routers to smooth out fluctuations in path quality along with delay-tolerant storage, to overcome total disconnection of the mobile device. The performance of the GSTAR protocol is evaluated for exemplary wireless access network scenarios using ns-3 based simulation models, and key design parameters are investigated.

Each node in GSTAR maintains two kinds of topology information. The intra-partition graph contains information about path quality between nodes in the current partition of the network. The path quality is determined using two metrics: short term and long term expected transmission time (SETT and LETT). Every node compares these two metrics using the store/forward decision threshold and stores the data on finding that the path is degraded with the expectation that it may improve in the future. Inter-partition graph gives a probabilistic view of the connection patterns between nodes in the network. It is used in the event of disconnections or partitions.

An ns-3 based simulation model is described which includes nodes with storage, hop-by-hop transport, time-varying wireless channels and mobile users with possible disconnection. The model is used to evaluate different forwarding algorithms in GSTAR. Using a baseline threshold scheme where packets are temporarily stored when $SETT > 1.1 \cdot LETT$, it is shown that the resulting system achieves performance improvements over the baseline with no storage. The threshold algorithm is studied further to consider adaptive settings based on the moving average and other temporal filters of the SETT sequence. The results show that if link quality fluctuations are random, the moving average scheme works well, while an exponentially weighted moving average is recommended for on-off channels with periodic outages. Simulation results are provided in each case, showing the benefit of adaptive threshold settings over the baseline non-adaptive case considered in earlier work.

Acknowledgements

This thesis is the result of constant support, encouragement and feedback of my advisor, Prof. Dipankar Raychaudhari. His objective advice and extraordinary technical knowledge enabled me to develop an understanding of the project.

I would like to express my sincere appreciation for Dr Samuel Nelson's contribution at the beginning of the project and in setting up the NS-3 simulation model.

I am heartily thankful to Ivan Seskar for patiently answering all my questions and helping me resolve all the issues that came up during my research. Without his help, it would have been very difficult to complete the thesis.

I also acknowledge the help of my friend Abhishek Chanda at all the stages of this project.

Table of Contents

List of Figures

Chapter 1

Introduction

The current internet architecture considers mobile devices as end hosts and not a part of core Internet. But, with the recent proliferation of wireless and mobile devices, the future Internet would be highly heterogeneous with no strict characterization of core vs access, wired vs wireless, ad-hoc vs managed etc. This brings about some interesting challenges like host and network mobility, variable link quality and connectivity, network partitions and multi-homing. The future Internet requires an efficient and robust support for handling these mobility related challanges. [1]

To this end, several clean-slate proposols have come up like [2, 3, 4, 5, 6, 7]. One such project is the NSF-funded Future Internet Architecture (FIA) project, called *MobilityFirst*. MobilityFirst treats the mobile devices and associated application as first-class Internet citizens.

1.1 MobilityFirst - Future Internet Architecture

Due to the hierarchical, aggregation-based nature of BGP, along with the current concept of IP addresses being used for higher-level naming, the current Internet is not flexible enough to handle mobility and its challenges.

The MobilityFirst architecture has the following key features:

- Each device is provided with both a name (called Globally Unique ID or GUID) and an address or addresses (in case of multi-homed devices).
- The address could be bound to the name any time during the transmision of a data packet. Thus, packets could be routed using either name or address. This late binding to an address enables the network to handle multi-homing, where

packets for a multi-homed device could be routed using only GUID.

- Each node in the network uses in-network storage with hop-by-hop transport of large data units to deal with all forms of mobility and the associated challenges in a unified manner.
- It provides efficient support for handling anycast and multicast data.

Detailed information on *MobilityFirst* architecture is provided in [8, 9, 10].

Since most of the interesting mobility related challenges will occur relatively close to end users, it is critical for future protocols to support a flexible, robust, and unified means of exchanging routing information in a local area through many different types of environment. To this end, we present GSTAR, a generalized storage-aware intra-domain routing protocol capable of high performance in a variety of mobility-driven environments, including wired, wireless mesh, wireless ad-hoc, and DTN. GSTAR is a local, or intra-domain, routing protocol based on the *MobilityFirst* architecture.

At a high level, GSTAR maintains time-sensitive information about links within its currently connected component (e.g., all nodes to which an instantaneous end-to-end path exists from the node in question) and time-insensitive information about general connection patterns between all nodes in the network. It attempts to use the time-sensitive information when possible, and fall back on the connection patterns when needed. In this way, it can be thought of as a MANET+DTN protocol that is easily extended to more stable, perhaps wired, environments.

1.2 Problem Statement

High mobility in the Internet can lead to complete disconnections, where end-to-end protocols such as TCP fail as they require a path to be setup before data is sent. A number of solutions have been proposed to overcome this problem, particularly from the *delay-tolerant networking* community. Techniques such as message replication in [11, 12, 13] and hop-by-hop transport [14] are utilized to bridge partitions in the network.

Unfortunately, there has been no comprehensive solution to bridge varying levels of connectivity, in that DTN protocols are usually not sufficient in highly connected environments and MANET protocols fail in highly disconnected environments.

While there has been some work on merging DTN and MANET protocols, they usually consider DTN nodes as specialized entities useful only for extending MANET protocols [15, 16], or consider MANET clusters to be relatively static and simply bridged by DTN nodes [17].

With GSTAR, we envision both DTN and MANET capabilities in all nodes, allowing them to appropriately choose techniques in a more fluid manner with no reliance on the stability of a local cluster.

For nodes that it has an instantaneous end-to-end path to, it makes both path selection and transmission decisions based on factors such as link quality and storage availability. It temporarily and proactively stores data when a problem is detected with the upstream path, in an attempt to not add to the congestion or make unnecessary retransmissions. Furthermore, if the destination is detected to be outside of partition, it will utilize connection probability information that is proactively disseminated throughout the network to progress the message, relying on routers' ability to store.

Thus, with GSTAR, all these mobility related challanges are handled at the network layer.

1.2.1 Previous Work: Cache and Forward

Cache and Forward (CNF) [18], [1] project was a clean-slate protocol that aimed to take advantage of cheap storage to deal with mobile traffic in the future Internet. It was designed to deal with variable link quality and connectivity inherent with the mobile devices.

In CNF, the link quality between any two mobile devices is determined using two metrics called short term estimated transmission time (SETT) and long term estimated transmission time (LETT). SETT is equal to the time required for transmission of a probe message between devices. LETT is computed as a simple average of past 10 SETTs.

Every node compares these two metrics to determine whether the current path quality is exceptionally good or bad. If the current path quality is bad as compared to the usual path quality, the node pro-actively stores the data in order for the links to get back to their normal state.

1.2.2 GSTAR

Each node running GSTAR maintains two kinds of topology information in its database. First is called the intra-partition graph which deals with the nodes in the current partition only. The other is inter-partition graph which gives a probabilistic view of the nodes in the complete network. Intra-domain routing table is based on CNF and Inter-domain routing table is based on a generalized single copy DTN routing protocol.

Intra-partition graph of GSTAR uses the same parameter settings of SETT and LETT as of CNF. But, SETT is computed by directly taking the current bit rate from the net-card. This gives the best possible transmission time.

1.2.3 Contributions

This thesis aims at evaluation of GSTAR protocol with different forwarding algorithms through NS3 based simulation.

We first set the forwarding logic to be same as that of CNF with LETT as an average of past 10 SETTs and store-forward comparison threshold equal to 1.1 and compare GSTAR with a traditional link-state protocol. It is observed that GSTAR provides considerable gain in throughput (or goodput) for both wireless and wired-wireless hybrid network environments.

Next, we explore various computation methods of LETT. We basically investigate two alternate scheme:

- exponentially weighted moving average with different weighting factors
- simple moving average with different amounts of past history

Lastly, we look at various ways of making the store or forward decision threshold dynamic to adjust automatically to the type of traffic and current network conditions.

1.3 Thesis Organization

The rest of the thesis is organized as follows. We explain the GSTAR protocol in Chapter 2. Chapter 3 gives an overview of the implementation of GSTAR in NS3. Chapter 4 presents some intial performance evaluation of GSTAR against a traditional link-state protocol augemented with storage. We further explore the parameter space of GSTAR with different forwarding algorithms in Chapter 5. Chapter 6 presents future work.

Chapter 2
GSTAR Protocol

2.1 Overview

Existing ad-hoc and DTN routing protocols handle some of the challanges inherent with mobile devices. But, a unified approach for dealing with all kinds of network environments within a future Internet framework has yet to be explored.

GSTAR is a proactive link state protocol, with added DTN capabilities. It is a combination of MANET and DTN routing techniques with in-network storage. Basically, GSTAR aims at overcoming the following three challenges associated with mobile devices:

- Fluctuations in link quality
- Varying levels of connections and disconnections
- Partitions in the network

Each node running GSTAR maintains two types of topology and path quality information. The first one, called intra-partition graph responds to link quality information of the nodes in the current partition. The second, called inter-partition graph responds to connection probabilities between all nodes in the network. These tables are formed by proactive dissemination of control messages. This enables all nodes to have an up-to-date view of the network topology both inside and outside of the node's current partition.

Intra-partition graph enables GSTAR to be sensitive to link quality fluctuations. On the other hand, inter-partition graph makes it robust enough to deal with network partitions and disconnections.

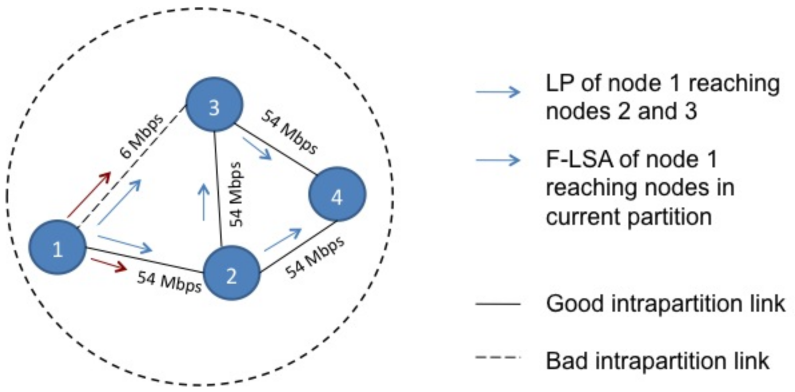

3
4
1
2
6 Mbps
54 Mbps
54 Mbps
54 Mbps
54 Mbps
LP of node 1 reaching nodes 2 and 3
F-LSA of node 1 reaching nodes in current partition
Good intrapartition link
Bad intrapartition link

Figure 2.1: GSTAR: Intra-partition Graph

The next hop for transmission of data is decided on the basis of link qualities and connection probabilities. Every node in the network is capable of storing the packet in the event of exceptionally bad link or disconnection in the route to the destination.

2.2 Intra-partition Graph

The intra-partition graph responds to time-sensitive information about the network topology. This graph maintains information about the quality of the link between all nodes in the current partition of network. The Expected Transmission Time (ETT) is used as the measure of link quality. This enables GSTAR to maintain an up-to-date view of the fine-grained changes in the network.

2.2.1 Control Messages

The intra-partition graph is formed using two types of control messages:

1. Link probe (LP)

2. Flooded link state advertisement (F-LSA)

The LP messages (Figure 2.1) are periodically broadcasted by evey node in the network to learn about their current one-hop neighbors. The LP message of Node 1 (shown in Figure 2.2) reaches nodes 2 and 3.

They are used in the following three ways:

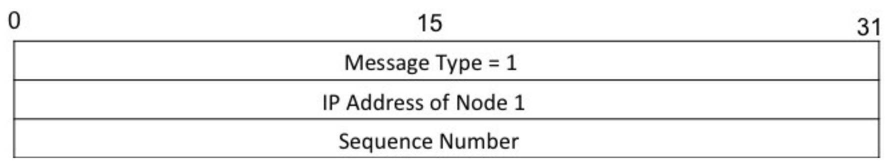


Figure 2.2: LP Packet of Node 1

- It allows the node to compute ETT for a given neighbor. For calculating the ETT, the node probes the data-link layer with the MAC Address of the neighbor for the transmission rate used for it. With this data-rate, ETT is set equal to the time required to transmit a 45 byte packet to the neighbor.

  $$ETT(msec) = \frac{45 \cdot 8 \cdot 1000}{datarate}$$

  where *datarate* is in Mbps.

- It enables each node to update a running database of contact probabilities for all nodes in the network. This is used for inter-partition graph (explained in next section).

The ETTs calculated through the link probes are used to estimate short term and long term link quality. These are called Short term Expected Transmission Time (SETT) and Long term Expected Transmission Time (LETT). SETT is computed by taking the average of last three ETTs. The computation of LETT can take two forms:

1. Exponentially weighted moving average (EWMA) of past SETTs

   $$LETT = \alpha \cdot SETT + (1 - \alpha) \cdot LETT$$

   where α is a weighting factor between $0 < \alpha < 1$

2. simple moving average of past SETTs

These are explored via simulation (as explained in Chapter 5).

In addition to link probe messages, each node periodically floods the network with F-LSA messages. F-LSAs carry one-hop neighbor information i.e., one-hop neighbor

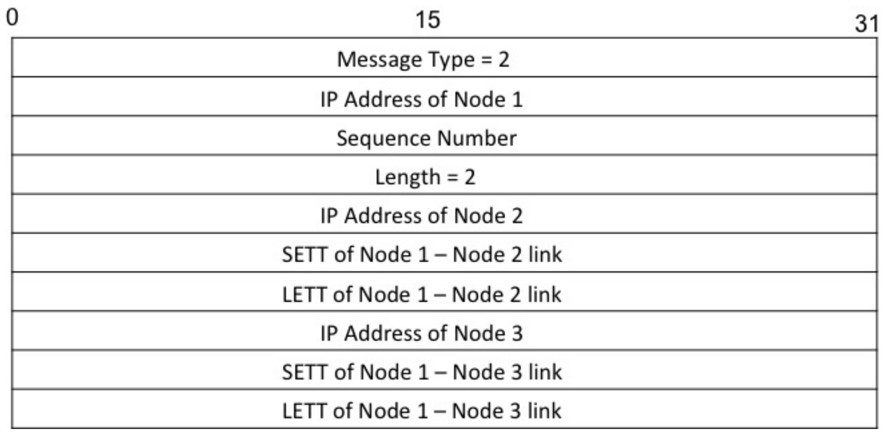

0
15
31
Message Type = 2
IP Address of Node 1
Sequence Number
Length = 2
IP Address of Node 2
SETT of Node 1 – Node 2 link
LETT of Node 1 – Node 2 link
IP Address of Node 3
SETT of Node 1 – Node 3 link
LETT of Node 1 – Node 3 link

Figure 2.3: F-LSA Packet of Node 1

address with its SETT and LETT. These messages are periodically broadcasted by every node in the network.

A node on receiving an F-LSA, updates its own database and then re-broadcastes the message. Thus, F-LSAs of a node are received by all the other nodes in a given network partition. This enables each node to have an up-to-date view of the current network topology called *intra-partition graph.*

In the network of Figure 2.1, an F-LSA of Node 1 contains the SETT and LETT for its neighboring nodes 2 and 3. This packet format is shown in Figure 2.3.

2.2.2 Intra-partition forwarding table

The F-LSA messages are used to compute the *intra-partition forwarding table*. This table provides the next hops for every other node to which an end-to-end path exists. The next hop corresponds to the route with lowest possible transmission time to the destination. These routes are computed by any shortest path algorthim, such as Djikstra's algorithm, using SETT values as weights. This sum of SETT values along the path is called *short term path quality* ($STPath$). In addition to $STPath$, *long term path quality* ($LTPath$) is computed as the sum of the LETT values along the selected path.

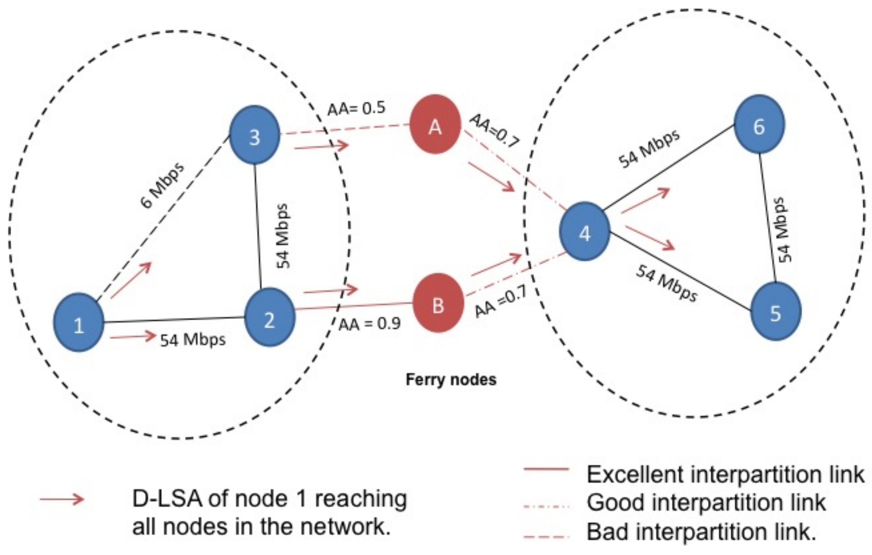


Figure 2.4: GSTAR: Inter-partition Graph

2.3 Inter-partition Graph

The inter-partition graph corresponds to time-insensitive information about general connectivity patterns for the entire network. This graph maintains information about the frequency or likelihood of a connection between nodes. The frequency or likelihood is represented in terms of connection probability. Thus, GSTAR is able to respond to the coarse-grain changes within the complete network.

2.3.1 Control Messages

The inter-partition graph is formed using the following control messages:

1. Link probe (LP)
2. Summary vector
3. Summary vector ACK
4. Disseminated link state advertisement (D-LSA)

The connection probability determines the percentage of time a node is connected to some other node in the network. Each node maintains a running database of the "on" time and "off" time for every other node it has come in contact (i.e. was a 1 hop neighbor anytime in the past). This is called “Connectivity Table”.“on” time signifies the amount of time the two nodes had an active connection, whereas, “off” time is the amount of time the nodes where disconnected. GSTAR periodically checks the neighbor table and updates the “on” and “off” time for all the nodes in the “Connectivity Table”.

The connection probability is computed using the “on ” time and “off” time. It is called average availability (AA) of a node and is given by:

$$AA = \frac{on}{on + off}$$

In order to bridge partitions in the network, these average availabilties should reach every other node in the complete network. This is accompalished through D-LSA messages. These messages are epidimecially disseminated. A received D-LSA is carried indefinately by all nodes running GSTAR. The main aim here is to make this connectivity patterns to reach all nodes in the entire network. This is carried out through Summary vector and Summary vector ACK messsages.

Every node periodically unicasts a Summary vector packet containing the summary of all available D-LSAs. The receiving node checks its own database for the available D-LSAs and replies with a summary vector ACK containing addresses of nodes whose D-LSAs are unavailable. On receiving the ACK, the corresponding D-LSAs are broadcasted.

As these D-LSAs are carried indefinitely, it is possible for a node to recieve the D-LSA of nodes that were never within its partition. This enables a node to know about all other nodes in the entire network.

Consider the network with two partitions and two ferry nodes (Node A and Node B) as shown in Figure 2.4. When Node 2 comes in contact with Node B, it sends a summary vector packet as in Figure 2.5. Node B on receiving this packet, replies with a summary vector ACK of Figure 2.6. (Here, we are assuming that Node B already has a D-LSAs of Node 2 and Node 3). Thus, Node 2 transmits the D-LSA of Node 1 (Figure 2.7) to Node B. In the same way, Node B transmits this D-LSA of Node 1 in the other partition consisting of Nodes 4, 5 and 6. Thus, D-LSA of Node 1 reaches all

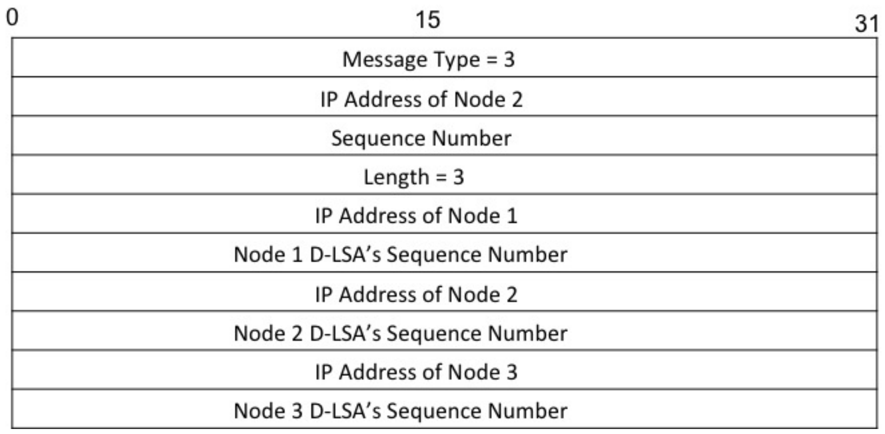


Figure 2.5: Summary Vector Packet of Node 2

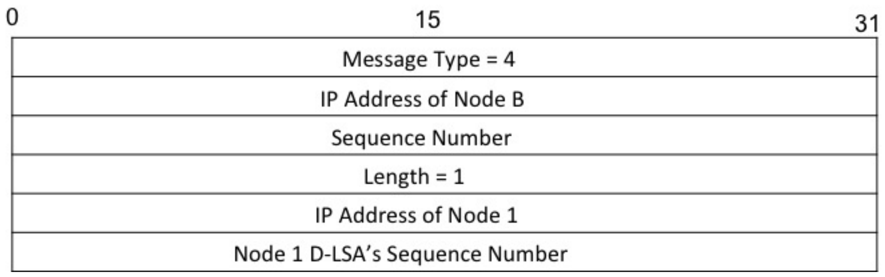


Figure 2.6: Summary Vector ACK Packet of Node B

the nodes in the complete network.

2.3.2 Inter-partition forwarding table

The D-LSA messages are used to compute the *inter-partition forwarding table.* This table provides the next hops for every other node in the entire network. The next hop corresponds to the route with best possible chance of meeting the destination. These routes are computed by any shortest path algorthim, such as Dijkstra's algorithm, using a function of the average availability as weights. GSTAR uses the approach taken by PREP [13] and defines a weight as $1 - AA + 0.01$, where the small constant is added to favor paths with shorter hop counts if the average availabilities are all close to 1.

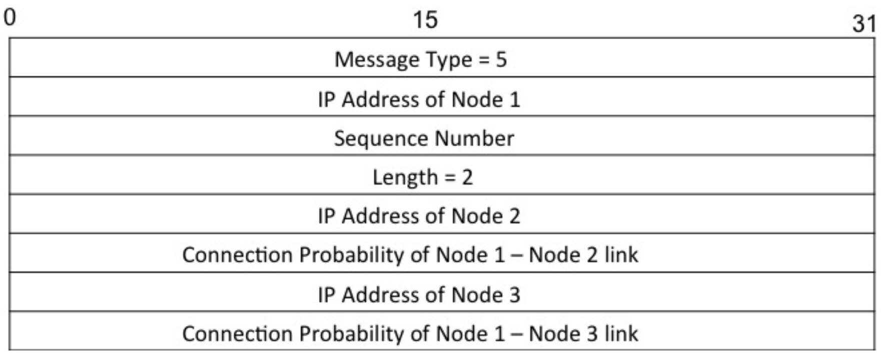


Figure 2.7: D-LSA Packet for Node 1

2.4 Transmission of Data

In a wireless network with varying levels of connectivity and partitions, it is not always possible to have a good end-to-end route to the destination. Thus, GSTAR provides hop-by-hop reliability for data packets. GSTAR combines a set of data packets into chunks. These chunks forms the autonomous unit of message transmission. All decisions related to routing including reliability are made at chunk level.

Assuming both forwarding tables have been computed, each node first looks for the destination node in its current network partition. It, therefore, checks its intra-partition table for an end-to-end path to the destination. Even though an end-to-end path exists, its possible for this path to be not good enough. This could be due to congestion or low signal-to-noise ratio anywhere along the route to the destination in the network. If a node finds the current path quality to be abormally bad, it proactively stores the chunk and waits for the path to get back to its original state. This decision is based on short and long term path qualities. A node stores the chunk, if it finds that

$$STPath > k \cdot LTPath$$

where k is a threshold which detects how bad the path must be, relative to its long term performance, before a decision to store the data is made. We explore this value via simulation (explained in Chapter 5).

This decision to store or forward is made at every node in the route to the destination. As, the F-LSAs are periodically flooded through the entire network partition, the

STPath and *LTPath* looks the same to all intermediate nodes in the path. Thus, once the node close to the source (or the source itself) finds the path to be of good quality, other nodes along the path would not generally store the data. They do store the data over the time it takes for the F-LSA of the node with the bad link to reach all the way down to the source.

Thus, GSTAR proactively stores data close to the source if upstream link quality issues are detected. This is advantageous for three reasons:

- If congestion is the cause of upstream link problems, this will not unnecessarily add to the congestion, instead waiting for it to resolve.
- Storing data away from faulty links in the network will minimize the number of lower-level retransmissions that must occur to push data through a poorly performing link, and hence not congest the area around the link.
- Storage around the faulty link has a higher chance of being depleted (due to routers around that area being forced to make store decisions), and hence adding to the storage problem in that area will hurt flows that must traverse that area, but not the faulty link itself.

If no end-to-end path exists for the destination in the intra-partition table; the node switches to DTN mode by checking its inter-partiton table. In this case, the node proactively pushes the message along the most probable route to the destination.

Thus, by considering whether or not the destination for a chunk is part of the node's current partition, the router will know how to appropriately respond. This may include proactively storing the message, if a path exists but is of low quality, or proactively pushing the message, if no instantaneous end-to-end path exists.

Chapter 3

Implementation of GSTAR

We implemented GSTAR with hop-by-hop transport in Network Simulator 3. Each node in the network is identified by a unique IP address. IP headers and link layer headers are appended to all the GSTAR control and data packets. This chapters provides an overview of the various databases maintained at every node in the network.

3.1 Implementation of Routing

3.1.1 Intra-Partition Graph

Each node in the network periodically broadcasts link probe message every 1 second. A node on receiving this link probe, probes its link layer with the IP address of the neighbor for the transmission rate. This transmission rate is used for calculation of ETT. Every node maintains a neighbor table with the IP address of the neighbor with the corresponding SETT and LETT. This table is updated everytime a new link probe is received. A node also maintains a timer of 5 seconds with each entry in the neighbor table. This timer is restarted on receiving a new link probe from the neighbor. If a link probe is not received for 5 seconds, it is assumed that the node is not reachable anymore amd the corresponding neighbor entry is deleted. The neighbor table at Node 1 for the network of Figure 2.4 is shown below.

The information in the neighbor table is encapsulated in the F-LSA packet and periodically broadcasted every 2 seconds. The F-LSAs are stored in F-LSA table. A node on receiving a F-LSA checks if this F-LSA is newer than the one it has in the F-LSA table. If this is of a higher sequence number than the one in its database, the node deletes the pervious F-LSA and saves this newly received F-LSA in its F-LSA table.

Neighbor	SETT	LETT
2	6666	6666
3	60000	60000

Figure 3.1: Neighbor Table at Node 1 for the network of Figure 2.4

Destination	Next Hop	ST Path	LT Path	No of Hops
2	2	13332	13332	1
3	2	66666	66666	2

Figure 3.2: Intra-Partition Forwarding Table at Node 1 for the network of Figure 2.4

At the same time, the node re-broadcasts this new F-LSA. Also, a timer of 5 seconds is started with every entry in the F-LSA table. This timer automatically deletes that F-LSA from the database after 5 seconds. Thus, the F-LSA table contains F-LSAs of the nodes in the current partition only.

The neighbor table and F-LSA table are used to compute intra-partition forwarding table [Figure 3.2]. It contains the next-hop IP address with the $STPath$ and $LTPath$. It is formed using Djikstra's algorthim with SETT as the link weights. This table is recalculated whenever a link probe or F-LSA is received or deleted.

3.1.2 Inter-Partition Graph

Each node updates the "Connectivity Table" every 1 second. This table conatins the average availabilty of the nodes. The information in the "Connectivity Table" is encapsulated in the D-LSA packet. The "Connectivity Table" at Node 2 is shown below.

Each node in the network periodically broadcasts summary vector message every 5 second to its neighbors. This message contains the summary of the D-LSAs available in D-LSA table. A node on receiving a summary vector, checks its own D-LSA table for any new D-LSAs available with the neighbor. It then replies with a summary vector ACK containing node addresses of these new D-LSAs. A node on receiving a summary vector ACK, broadcastes the requested D-LSAs.

All these D-LSAs are saved in D-LSA table. The entries in the D-LSA table are carried indefinately. Thus, they help to bridge partitions in the network.

The connectivity and D-LSA tables are used to compute inter-partition forwarding

Neighbor	Connection Probability
1	1
3	1
B	0.9

Figure 3.3: Connectivity Table at Node 1 for the network of Figure 2.4

Destination	Next Hop	AA
2	2	0.01
3	3	0.02
A	3	0.51
B	B	0.01
4	B	0.42
5	B	0.43
6	B	0.43

Figure 3.4: Inter-Partition Forwarding Table at Node 2 for the network of Figure 2.4

table [Figure 3.4]. This table is formed using Djikstra's algorthim with $1 - AA + 0.01$ as the link weights. This table is recalculated whenever a new D-LSA is received.

3.2 Implementation of Hop-by-Hop transport

The autonomous unit of transmission at network layer is a chunk. A chunk consists of CHK PKT COUNT number of data packets. Every node acknowledges a reception of a chunk to the neighbor sending it.

Every node maintains following two databases:

- DataSet: A Data tuple consists of a Chunk ID, Source IP address and all the data packets for that chunk. DataSet is a collection of such Data tuples.
- BitmapSet: A Bitmap tuple consists of a Chunk ID, Source IP address and a bitmap for that chunk. A bitmap is an array of size CHK PKT COUNT bytes, each position corresponding to a data packet. BitmapSet is a collection of such Bitmap tuples.

The source Node A (refer Figure 3.5) first checks its forwarding tables to find the next-hop IP address and the corresponding $STPath$ and $LTPath$. It then checks to see

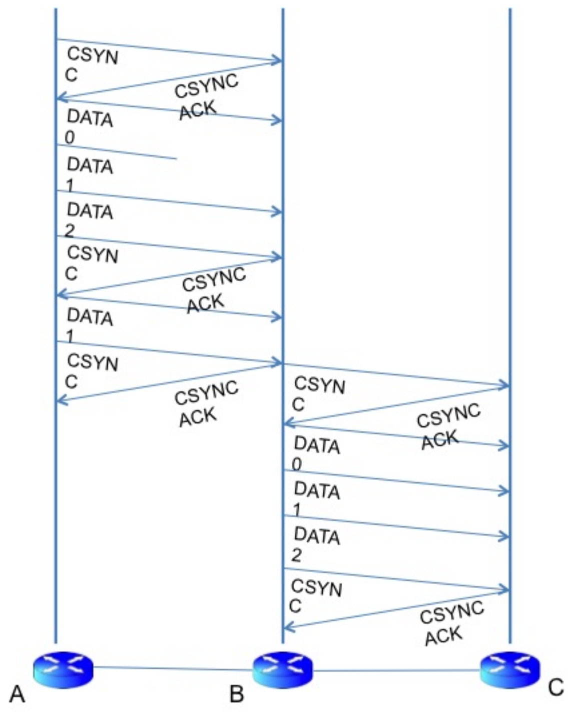
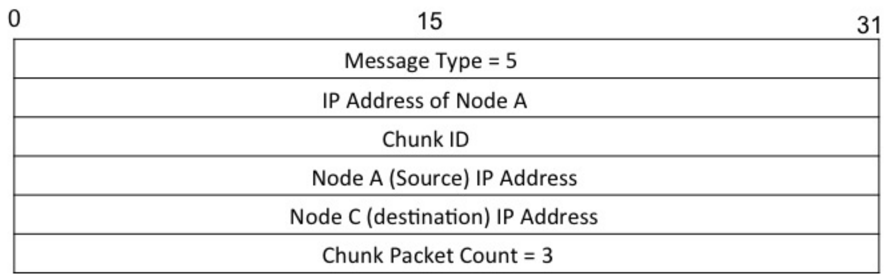


Figure 3.5: Hop by hop Data Transfer with CHK PKT COUNT = 3



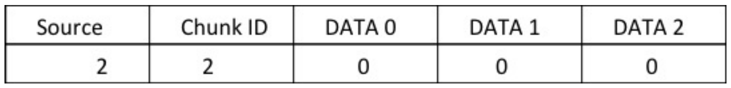
Message Type = 5
IP Address of Node A
Chunk ID
Node A (Source) IP Address
Node C (destination) IP Address
Chunk Packet Count = 3

Figure 3.6: CSYNC Packet by Node A

if the path quality is reasonable enough by comparing these two metrics. If the path is found to be of good quality, it starts the transmission by sending a CSYNC packet [Figure 3.6] to the next-hop.

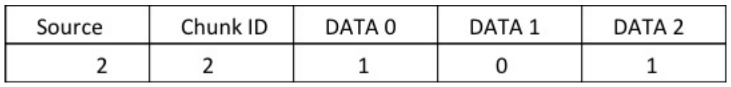
On receiving a CSYNC, the node checks its BitmapSet for the corresponding source and chunk ID. If there isn't an entry in BitmapSet, it creates one and initilizes the the bitmap array to 0. This bitmap is sent back as a CSYNC ACK to the source.

The source on receiving the CSYNC ACK, checks the bitmap and transmits the data packets for which the bitmap conatins a 0. Thus, in the Figure 3.5, all three data pakets in the chunk are transmitted. Also, it resends the CSYNC.



Source	Chunk ID	DATA 0	DATA 1	DATA 2
2	2	0	0	0

Figure 3.7: Bitmap Tuple at Node B after receiving first CSYNC packet



Source	Chunk ID	DATA 0	DATA 1	DATA 2
2	2	1	0	1

Figure 3.8: Bitmap Tuple at Node B before receiving second CSYNC packet

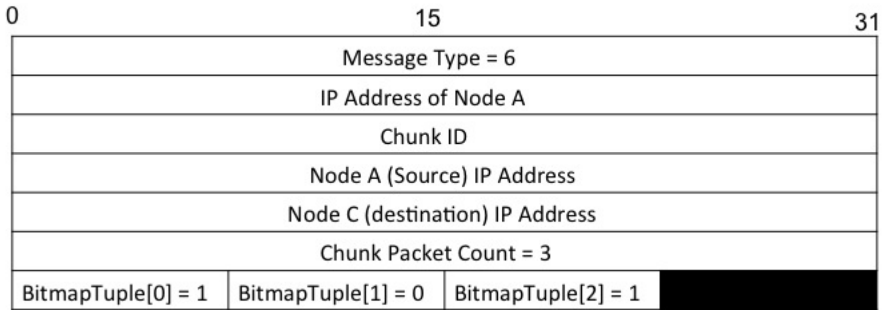
A node on receiving a data packet, creates a Data tuple and saves it in the DataSet. At the same time, it sets the corresponding position in the bitmap array of the Bitmap tuple to 1. On receiving the second CSYNC, this modified bitmap tuple is encapsulated in the CSYNC ACK and forwarded to the source.

If all the positions in the bitmap of CSYNC ACK is 1, the source starts the transmission for the next chunk in the same way. If some entries are still set to 0, it re-transmits those data packets and again sends the CSYNC. Thus, Data packet with sequence number 1 is re-transmitted in Figure 3.5.

Every node on receiving a complete chunk, checks the destination address in it. If the destination address is same as its own IP address, the chunk is moved to the application. If that chunk is destined for some other node, it checks the forwarding tables to find the next-hop and the same process is repeated. As a result, in Figure 3.5, Node B after receiving the third CSYNC, starts transmitting the chunk to Node C.

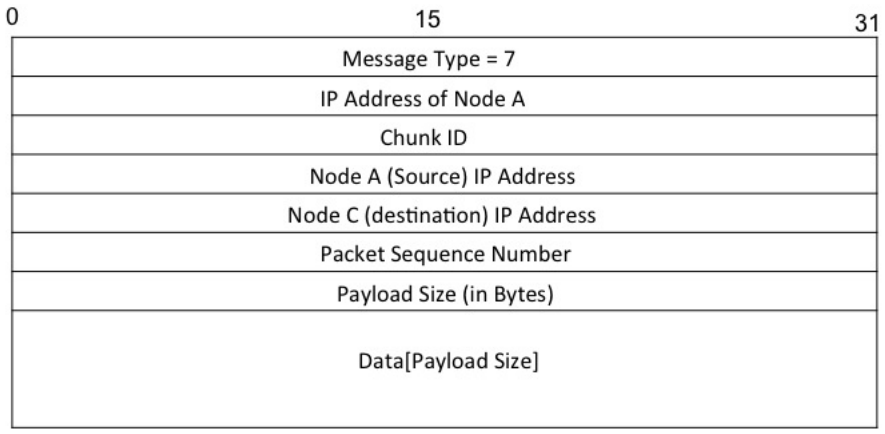
If any node does not receives a CSYNC ACK for its CSYNC in 5 seconds, the CSYNC is re-transmitted.

The current implementation of hop-by-hop transport provides each node with infinite storage. This is to ensure that no packets are ever dropped by GSTAR layer.



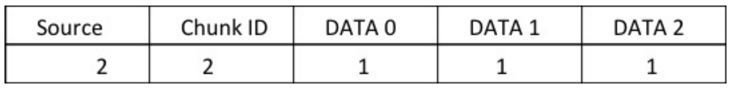
0	15	31
Message Type = 6		
IP Address of Node A		
Chunk ID		
Node A (Source) IP Address		
Node C (destination) IP Address		
Chunk Packet Count = 3		
BitmapTuple[0] = 1	BitmapTuple[1] = 0	BitmapTuple[2] = 1	■

Figure 3.9: CSYNC ACK Packet by Node B



0	15	31
Message Type = 7		
IP Address of Node A		
Chunk ID		
Node A (Source) IP Address		
Node C (destination) IP Address		
Packet Sequence Number		
Payload Size (in Bytes)		
Data[Payload Size]		

Figure 3.10: DATA Packet by Node B



Source	Chunk ID	DATA 0	DATA 1	DATA 2
2	2	1	1	1

Figure 3.11: Bitmap Tuple at Node B before receiving third CSYNC packet

Chapter 4

Performance Evaluation of GSTAR

The first set of simulations demonstrate the effectiveness of proactive storing in case of abnormally bad links somewhere along the route to the destination. We perform a comparative study of GSTAR and a traditional link state protocol augmented with a hop-by-hop transport protocol.

We explore complete wireless, mixed wired-wireless (hybrid) and disconnected network environments. It is observed that GSTAR provides considerable gain in throughput for all these networks.

In these simulations, we set the LETT to be an average of past 10 SETTs. Also, the store or forward decision threshold is held static at 1.1. Thus, a node decides to store a chunk if

$$STPath > 1.1 \cdot LTPath$$

We consider goodput as the performance metric in all of the evaluations. It is measured as total number of chunks received by the destinations.

4.1 GSTAR vs Traditional Link State protocol in Wireless network

We consider a multi-hop network of nine nodes (see Figure 5.10 for topology) each with a 802.11 net-card. The bit-rate for the links is set to 54 Mbps.

In this simulation setup we have two 5-hop flows from Node 1 to Dest 1 and Node 2 to Dest 2. The two sources continuously transmit chunks of 25 packets. The bit-rate of the Node 6 - Dest 1 link is periodically fluctuated from 54 Mbps to 6 Mbps. This simluates congestion in the network for one of the flows.

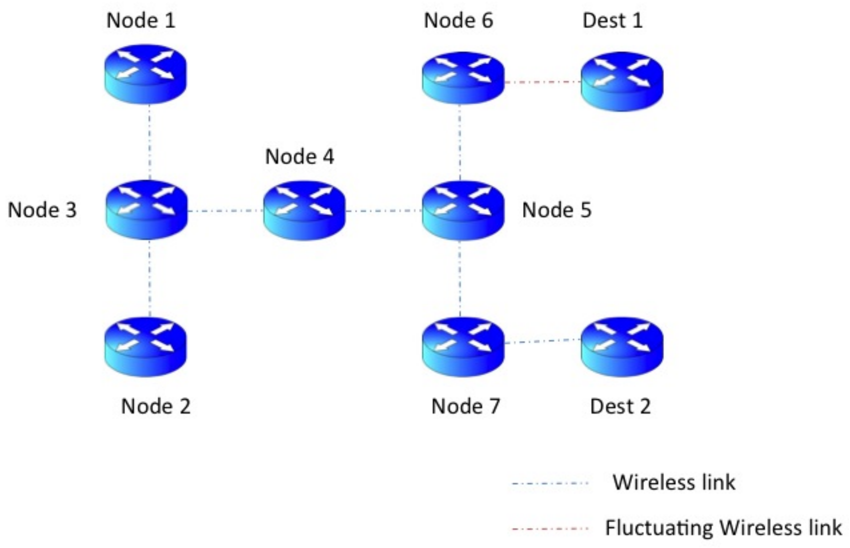


Figure 4.1: Wireless network topology

The bit-rate fluctuation period is varied from 20 + rand(-10/4,10/4) seconds to 40 + rand(-20/4,20/4) seconds with an increment of 5 seconds and equal on and off time. Each simulation is run for 90 second. Each data point in the graph (Figure 4.2) is an average of 10 runs. We also show 95% confidence interval for each point.

It is observed that GSTAR outperforms traditional link-state protocol with a considerable gain in aggregate goodput. With a traditional link-state protocol, congested Flow 1 and normal Flow 2 compete with each other to gain access to the channel. But, with 6 Mbps bit-rate, Flow 1 keeps this access for a longer duration than Flow 2. Thus, congestion in one flow affects the other flow in the network reducing overall network goodput. SETT and LETT enables GSTAR to detect this variation in link quality. Thus, it efficiently alleviates the effect of congestion in Flow 1 from Flow 2 resulting in better network utilization.

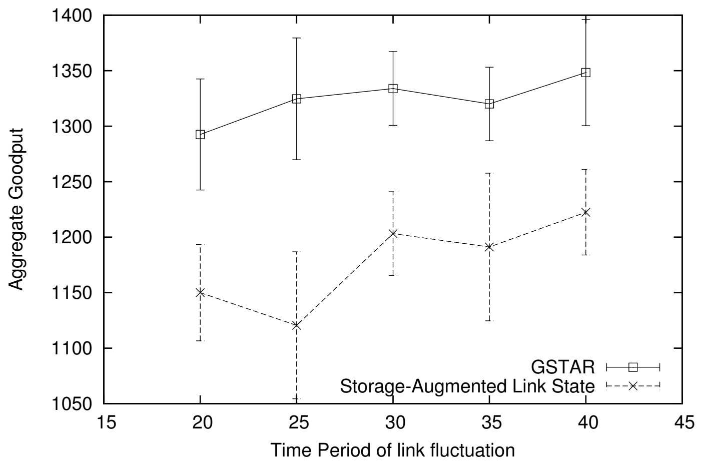


Figure 4.2: GSTAR vs Traditional Link State with Wireless network topology

4.2 GSTAR vs Traditional Link State protocol in Hybrid network

We consider a multi-hop network of nine nodes (see Figure 4.3 for topology). The links between Node 6 - Dest 1 and Node 7 - Dest 2 are wireless (802.11) and remaining are all wired (point to point) links. Thus, Node 6 and Node 7 are multi-homed. The bit rate for both wired and wireless links is set to 54 Mbps.

For this hybrid network, we run 2 types of simulation. In the first simulation, we have 4 flows:

- Node 1 - Dest 1 of 5 hops
- Node 2 - Dest 2 of 5 hops
- Node 3 - Dest 1 of 4 hops
- Node 4 - Dest 2 of 3 hops

The other simulation is with 6 flows:

- Node 1 - Dest 1 of 5 hops
- Node 2 - Dest 2 of 5 hops
- Node 3 - Dest 1 of 4 hops

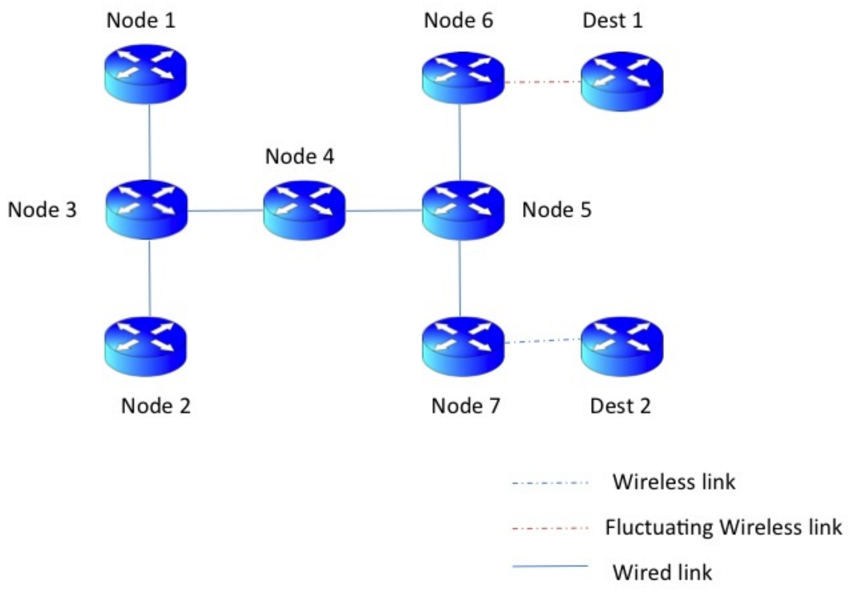


Figure 4.3: Hybrid network topology

- Node 4 - Dest 2 of 3 hops
- Node 5 - Dest 1 of 2 hops
- Node 6 - Dest 2 of 1 hop

For both the simulation setups, the bit-rate of the Node 6 - Dest 1 link is periodically varied from 54 Mbps to 6 Mbps. The fluctuation period is set to 30 + rand(-5,5) seconds with equal on and off time. The chunk size is set to 10 data packets. In these simulations, we vary the offered load by increasing the number of chunks generated per second by the sources. We start with 25 chunks per second by each source, followed by 50 to 200 in increments of 50. Each simulation is run for 90 second. Each data point in the graphs (Figure 4.4 and Figure 4.5) is an average of 10 runs. We also show 95% confidence interval for each point.

It is observed that GSTAR provides gain in aggregate goodput for medium to high offered load. This is because the congestion in one part of the network doesn't affect other parts as shown in the previous result. The cross-over points in the graphs are

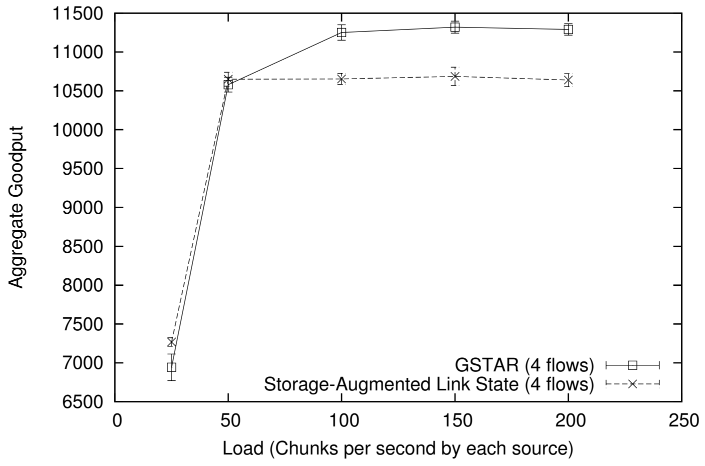


Figure 4.4: GSTAR vs Traditional Link State with Hybrid network topology (4 Flows)

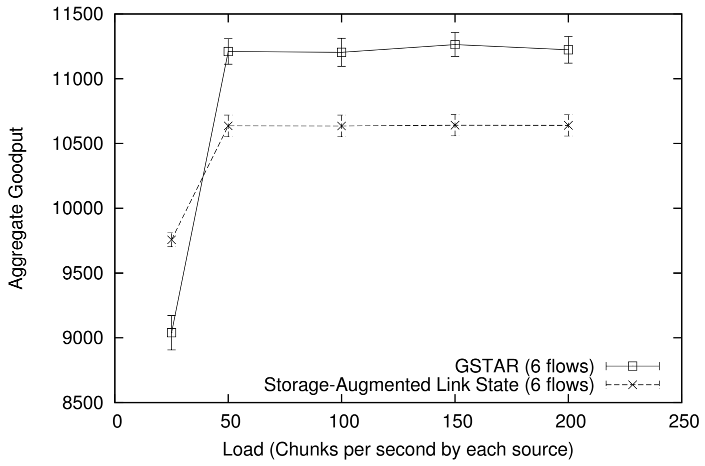


Figure 4.5: GSTAR vs Traditional Link State with Hybrid network topology (6 Flows)

the load at which network is fully utilized. At the points below complete utilization of network, the flows don't have to compete to gain access of the channel. Thus, storing chunks, while the channels are not utilized reduces the goodput from the network.

4.3 GSTAR with DTN vs GSTAR without DTN in Disconnected network environment

We consider the same hybrid muti-hop network topology of Figure 4.3. But, here the Dest 1 and Dest 2 are periodically disconnected from the remaining network as shown in

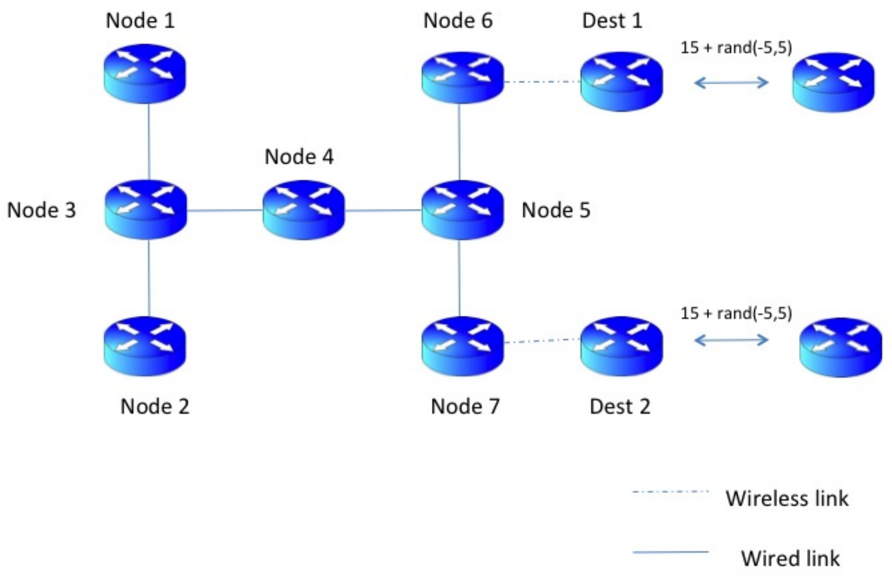


Figure 4.6: Disconnected hybrid network topology

Figure 4.6. The connection and disconnection period is set to 15 + rand(-5,5) seconds. In this simulation setup we have two 5-hop flows from Node 1 to Dest 1 and Node 2 to Dest 2. The chunk size is set to 25 packets. The bit rate for both wired and wireless links is 54 Mbps.

We vary the offered load by starting with 25 chunks per second by each source, followed by 50 to 200 in increments of 50. Each simulation is run for 90 second. Each data point in the graph (Figure 4.7) is an average of 10 runs. We also show 95% confidence interval for each point.

It is seen from the graph that for all types of offered load, GSTAR with DTN outperforms GSTAR without DTN providing a gain in aggregate goodput. GSTAR with DTN uses probabilistic view of the network to push the data further down the path. This proactive pushing enables the disconnected node to start receiving the data as soon as it reconnects with the complete network. Without DTN, the disconnected nodes have to wait for their F-LSAs to be received by the sources.

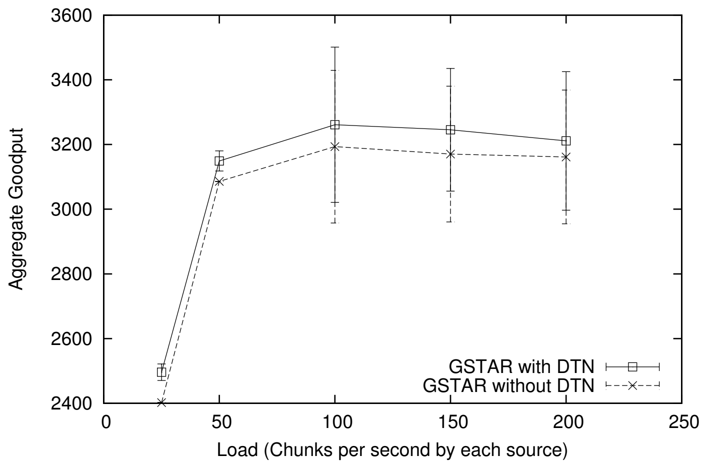


Figure 4.7: GSTAR with DTN vs GSTAR without DTN

4.4 Conclusion

These results emphasize that GSTAR as a whole, with its flooded F-LSAs and epidemic dissemination of D-LSAs, is able to detect different mobility challenges such as whether the node is disconnected or only the current link quality is bad. GSTAR augmented with storage enables the nodes to make intelligent proactive decisions to store or push the data depending on network conditions, resulting in better performance.

Chapter 5

Forwarding Algorithms for GSTAR

The previous results showed that even with a standard forwarding algorithm with LETT as simple moving average of past 10 SETTs and 1.1 as the store or forward decision threshold, storage aware routing performaned quite well. In this chapter, we look at different forwarding algorithms for GSTAR. This parameter exploration of GSTAR is divided into two parts. First, we look at different ways of computation of LETT. Next, we consider making the store or forward decision threshold adaptive to depend on network conditions.

5.1 Computation of LETT

The LETT can be computed in two ways:

- Exponentially weighted moving average (EWMA) of past SETTs:

  $$LETT = \alpha \cdot SETT + (1 - \alpha) \cdot LETT$$

  where α is a weighting factor explored via simulation

- Simple moving average of past SETTs, where we explore using different amounts of past history through simulation

For all the simulations, we consider the same multi-hop hybrid network of nine nodes (reproduced in Figure 5.1 for topology) where Node 6 and Node 7 are multi-homed. The bit rate for both wired and wireless links is 54 Mbps.

We have two 5-hop flows from Node 1 to Dest 1 and Node 2 to Dest 2 each transmiting chunks of 25 packets. To simulate congestion in the network, the bit-rate of the Node 6 - Dest 1 link is periodically fluctuated from 54 Mbps to 6 Mbps. The fluctuation period is set to 30 + rand(-5,5) seconds with equal on and off time.

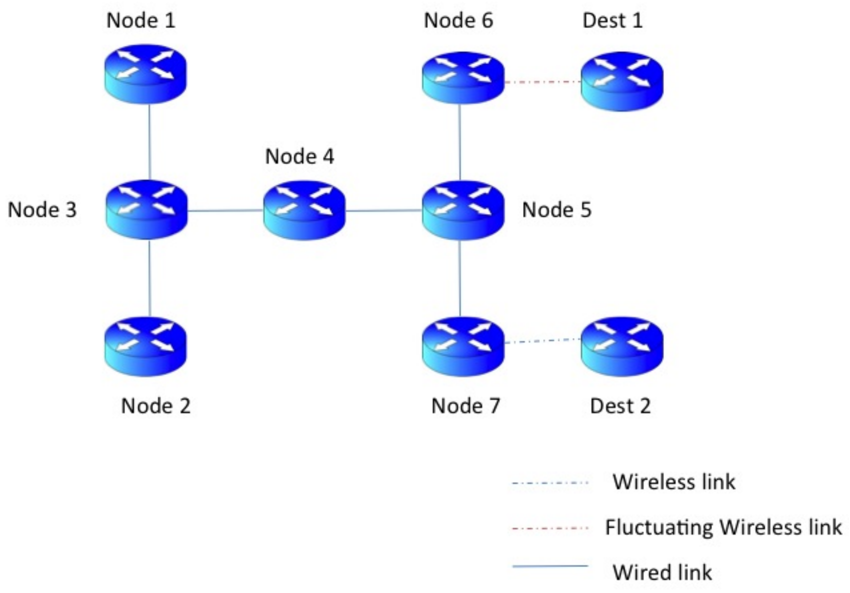


Figure 5.1: Hybrid network topology

5.1.1 EWMA: Performance Evaluation

This simulation compares EWMA with simple moving average of past 10 SETTs. We consider two cases of exponentially weighted moving average; one with α set to 0.1 and the other with α set to 0.5.

In this simulation setup, the offered load is varied by increasing the number of chunks generated per second by the two sources. We start with 25 chunks per second by each source, followed by 50 to 200 in increments of 50. Each simulation is run for 90 second. Each data point in the graph (Figure 5.2) is an average of 10 runs. We also show 95% confidence interval for each point.

It is observed that we get the best performance by giving more weights to past values of SETT. This is because our link quality variation method folllows a simple on-off model. Due to this periodic fluctuations, the past information of link quality is relevant in this network topology.

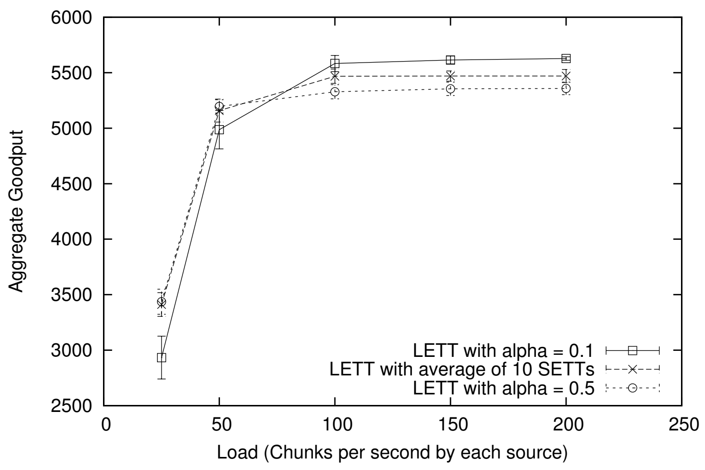


Figure 5.2: Computation of LETT with EWMA

5.1.2 Simple Moving Average: Performance Evaluation

This simulation set-up compares simple moving average with different amounts of past history. We consider compuatation of LETT using past 10 to 50, in increments of 10, SETTs.The offered load is set to 200 chunks per second by each source. Each simulation is run for 150 second.

It is observed from Figure 5.3 that the goodput from the network is at its maximum with using past 30, 40 and 50 SETTs. With 15 seconds of on and off period for the link; by giving equal weights to anywhere between 30 to 50 past SETTs, the LETT contains all the relevant information.

5.1.3 Conclusion

The best way for computation of LETT depends on the general trend in variation of link quality. Usually, this variation in links is not periodic and even if it is, the period varies with time. Hence, the weights for EWMA is usually difficult to predict for practical networks.

The other possible solution is to use Simple Moving Average with sufficient past history; so that LETT follows SETT slowly. As we saw in the previous results, using 30 seconds of past history seems reasonable enough if the link is bad for around 15 to

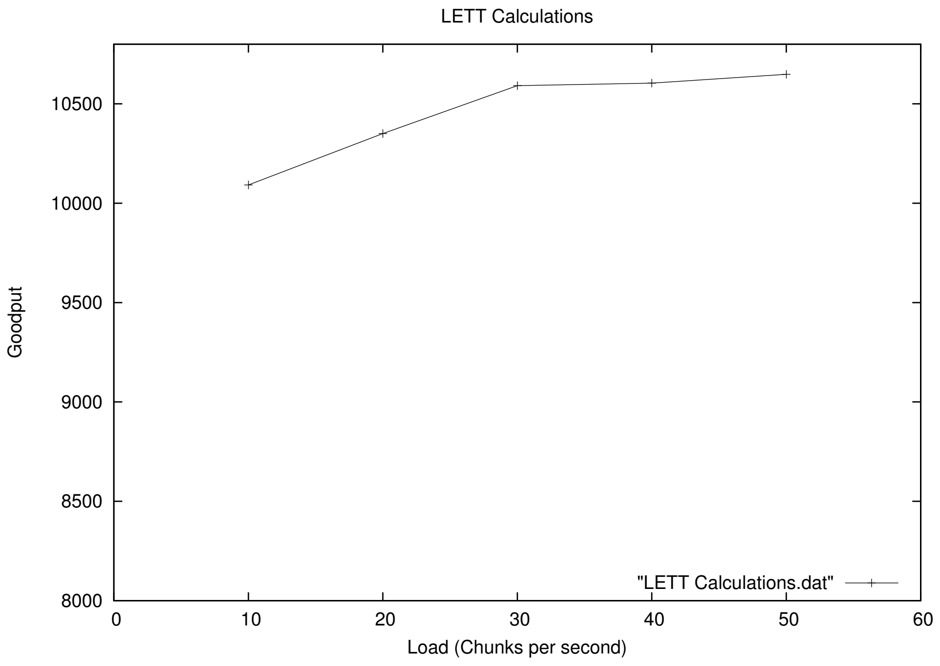


Figure 5.3: Computation of LETT with Simple Moving Average

20 seconds. This time is reasonable enough for the link to get back to its original state. Also, storing for longer than 20 seconds at a stretch is anyways not desirable.

5.2 Dynamic Store or Forward Decision Threshold

The previous results showed that static threshold of 1.1 with simple moving average of past 30 seconds history works well with on-off model and with some minor link fluctuations.

But, for the practical networks, in addition to short term and long term link quality, we need some way to detect the trend in link fluctuation. This trend in link fluctuation needs to be the basis for deciding the store or forward decision threshold. Instead of making it static at 1.1 which works well if the LETT follows SETT quickly, we need to make it dynamic, adapting itself to the current link fluctuation rate.

5.2.1 Approaches

We consider three possible ways of making this threshold dynamic. Each of these are some simple filtering techniques applied on past $STPath/LTPath$ ratios.

1. Simple Moving Average Filtering: Just as our intial computation of LETT, we take the average of past 10 $STPath/LTPath$ ratios. This averaged ratio value is then used as the threshold.

2. Median Filtering: Here, we take the median of past 10 $STPath/LTPath$ ratios. This median ratio value is then used as the threshold.

3. Moving Average + Median Filtering: This is a combination of past two techniques. First, we take a simple moving average of past 5 $STPath/LTPath$ ratio. Next, we perform median filtering on 5 such averaged ratio values. This technique, thus, takes a bit more past history into consideration.

Each of these techinques sets the minimum threshold to be 1.1. Thus whenever the ratio of $STPath/LTPath$ goes below 1.1, the store or forward decision threshold is set to 1.1.

5.2.2 Theoretical Analysis

We consider two data rate traces of 802.11 link between a moving car equipped with omnidirectional antenna and a base station; one in an apartment complex and the other in a parking lot [19]. We are using the data rates at every 1 second interval.

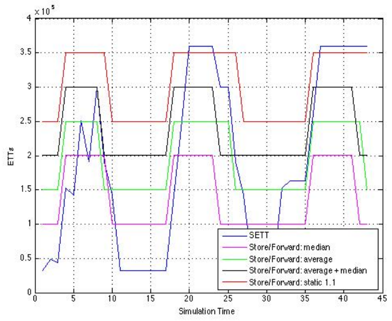


Figure 5.4: Store and Forward Regions with 802.11 link in apartment complex

Figure 5.4 and Figure 5.5 shows the the store and forward regions for these two links. It is observed that, theoretically, the averaging technique yields the most optimized result in both the cases. The other two techniques provides nearly the same results; but in some instances, the threshold rises too quickly and hence the forwarding starts

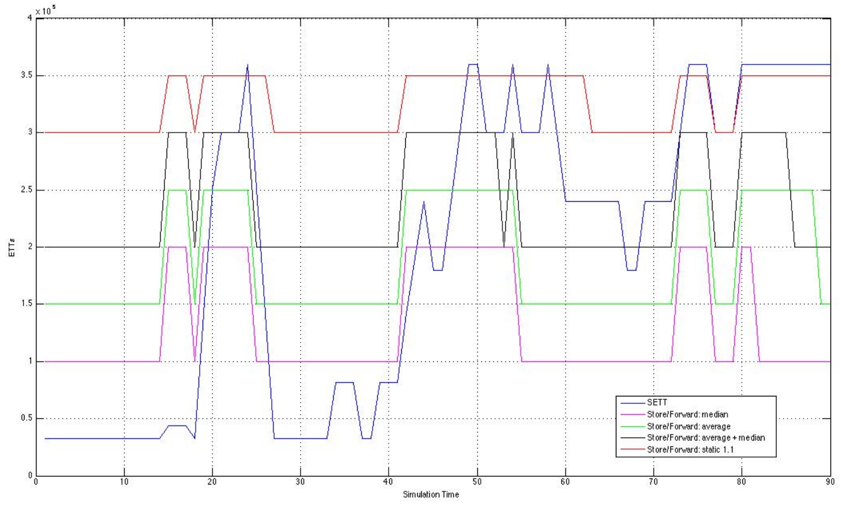


Figure 5.5: Store and Forward Regions with 802.11 link in parking lot

early. The static threshold of 1.1 works exactly the opposite by storing for a longer time.

This is also seen from the actual store or forward thresholds shown in Figure 5.6 and Figure 5.7 for both the cases. It is observed that the threshold with the averaging technique is the smoothest where as the other two techniques have sharp jumps. Thus, average filtering provides better results.

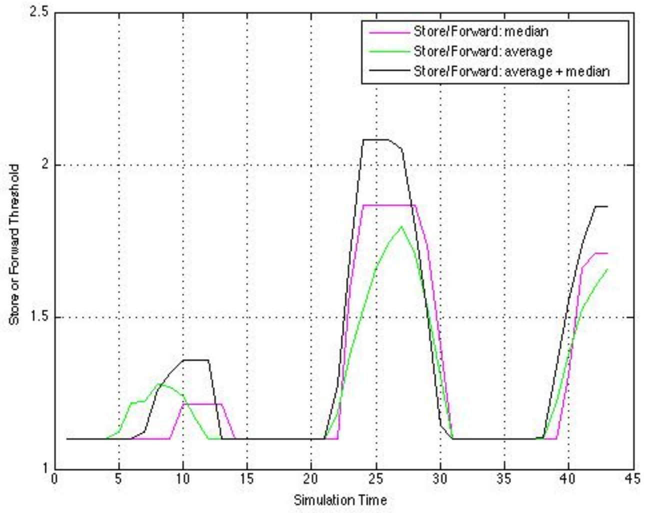


Figure 5.6: Store or Forward Decision Threshold for 802.11 link in apartment complex

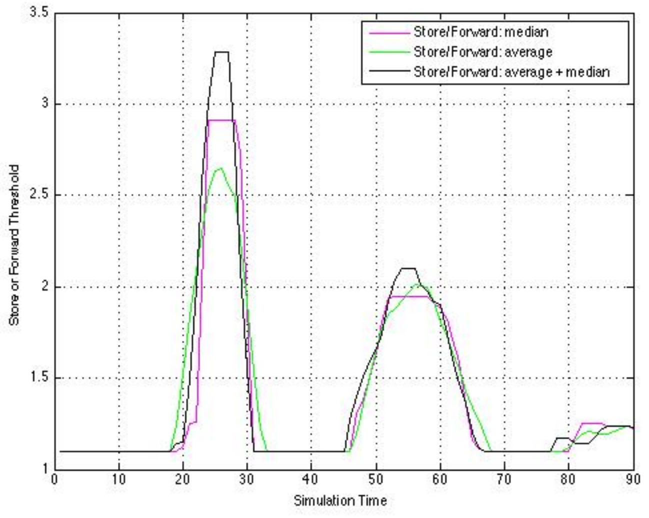


Figure 5.7: Store or Forward Decision Threshold for 802.11 link in parking lot

But for an on-off link fluctuation model with 30 seconds period, the static threshold of 1.1 works well as compared to a dynamic store-forward decision threshold. This is shown in Figure 5.8 where with the static threshold of 1.1, the data is stored for complete duration of off time.

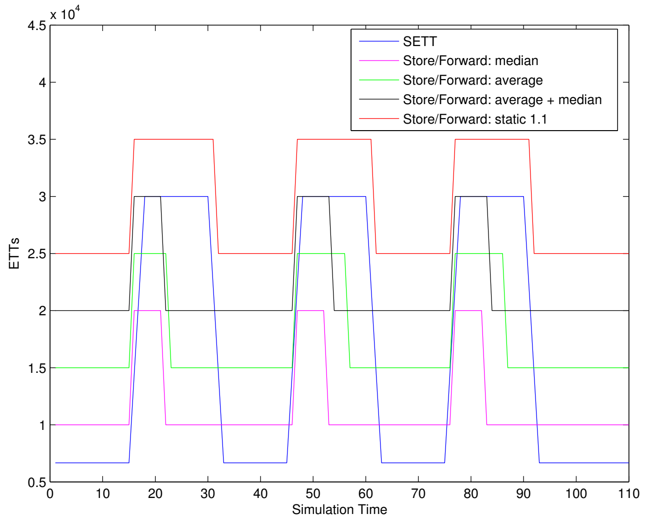


Figure 5.8: Store and Forward Regions with 802.11 link in on-off model

5.2.3 Simulation Results

We analyze the above three approaches of dynamic store or forward decision thresholds and compare them to the static threshold of 1.1 for multi-hop networks through ns3 simulations.

First is the same multi-hop hybrid network of nine nodes (see Figure 5.9 for topology) but here Node 1 and Node 2 are multi-homed. The bit rate for wired links is again set to 54 Mbps. The wireless links follow the trend of the previous two environments of apartment complex and parking lot.

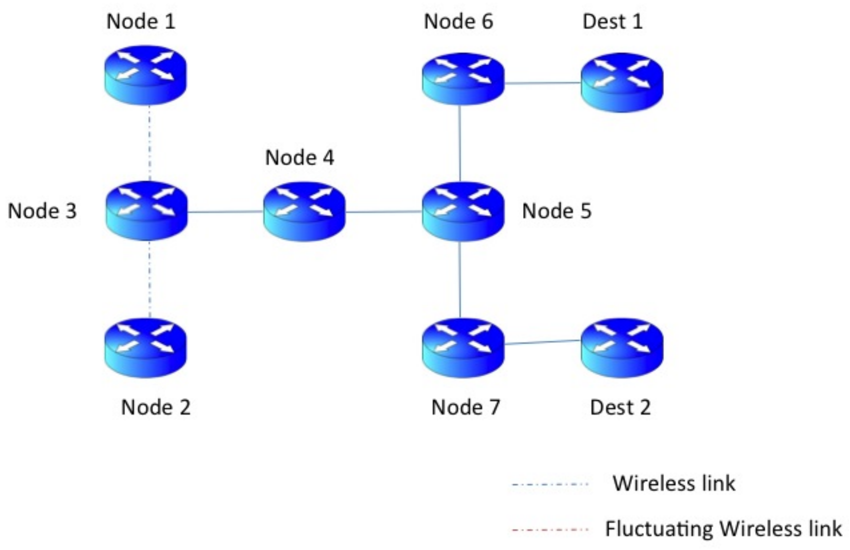


Figure 5.9: Hybrid network topology with wireless sources

The other is pure wireless networks as shown in Figure 5.10 where the Node 3 - Node 5 and Node 4 - Node 6 links fluctuate as per the data rate traces and remaining links are set to 54 Mbps.

The last set of simulation uses the nine nodes hybrid network topology of Figure 5.11 with Node 6 - Dest 1 and Node 7 - Dest 2 links as wireless. The bit rate for both wired and wireless links is set to 54 Mbps. The bit-rate of the Node 6 - Dest 1 link is periodically fluctuated from 54 Mbps to 6 Mbps. The fluctuation period is set to 30 +

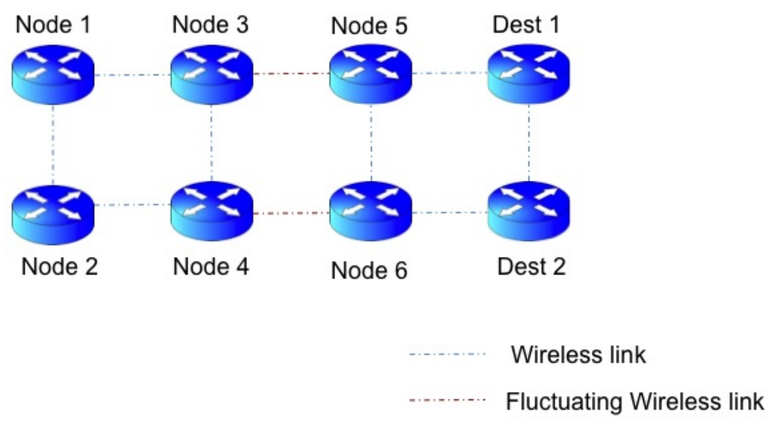


Figure 5.10: Wireless network topology

rand(-5,5) seconds with equal on and off time.

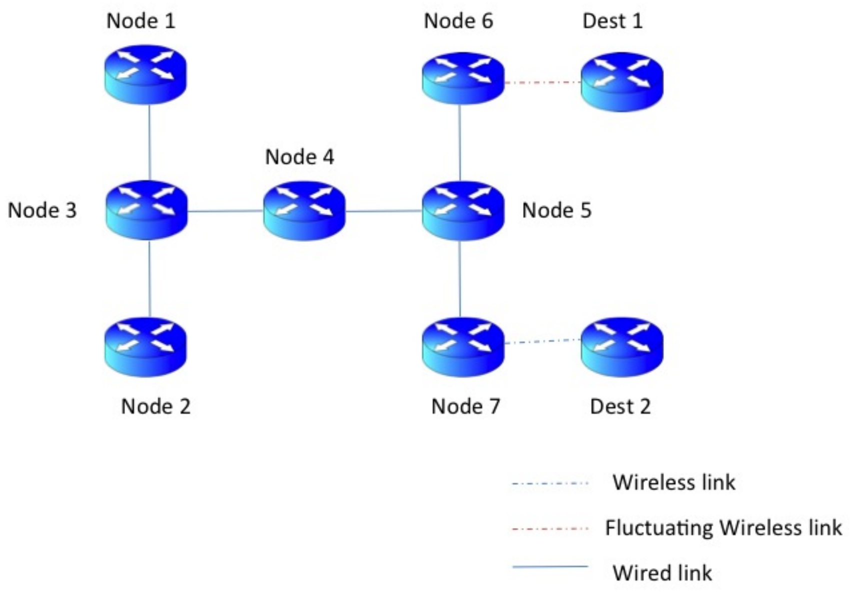


Figure 5.11: Hybrid network topology with wireless destinations (on-off model)

For all the networks, we have two flows from Node 1 to Dest 1 and Node 2 to Dest 2 each transmiting chunks of 25 packets. In these simulation setup, the offered load is varied by increasing the number of chunks generated per second by the sources. We start with 25 chunks per second by each source, followed by 50 to 200 in increments of 50. Each simulation is run for 45 second.

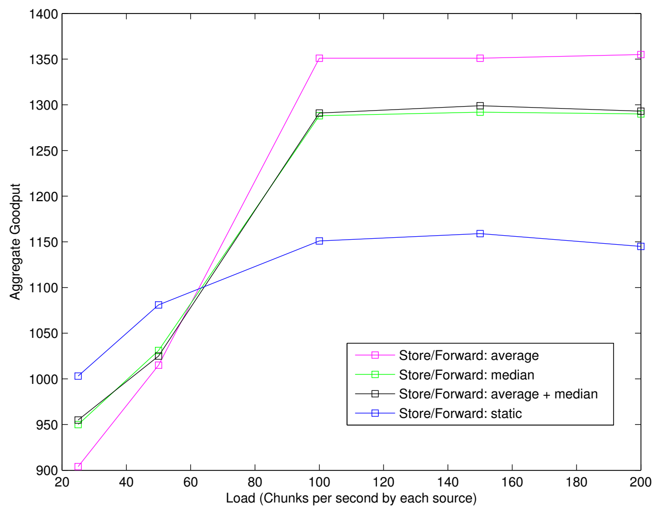


Figure 5.12: Dynamic Store or Forward Decision Threshold: Simulation Results for Hybrid network with wireless sources

It is observed from Figure 5.12 and Figure 5.13 that for medium to high offered loads averaging outperforms all other techniques, where as the static threshold of 1.1 provides the least goodput. But for on-off periodic fluctuation model, static threshold of 1.1 outperforms all other techniques (Figure 5.14) .

5.2.4 Conclusion

Thus, a static threshold of 1.1 works well if the link fluctuation is periodic. But if the link fluctuation is random then using a dynamic store or forward decision threshold provides better performance from the network.

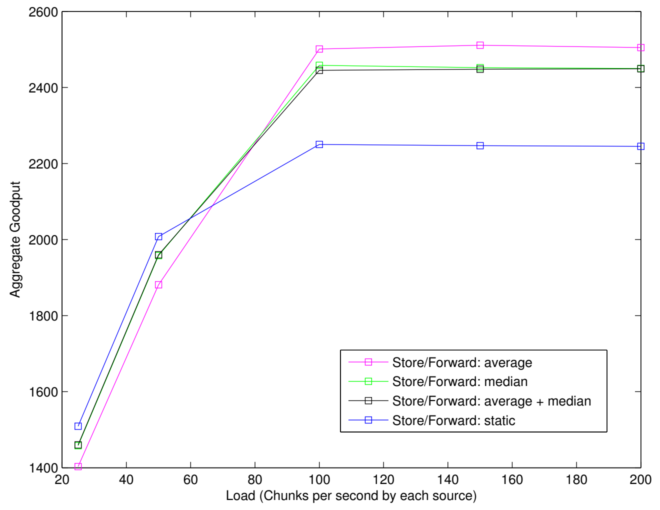


Figure 5.13: Dynamic Store or Forward Decision Threshold: Simulation Results for Wireless network

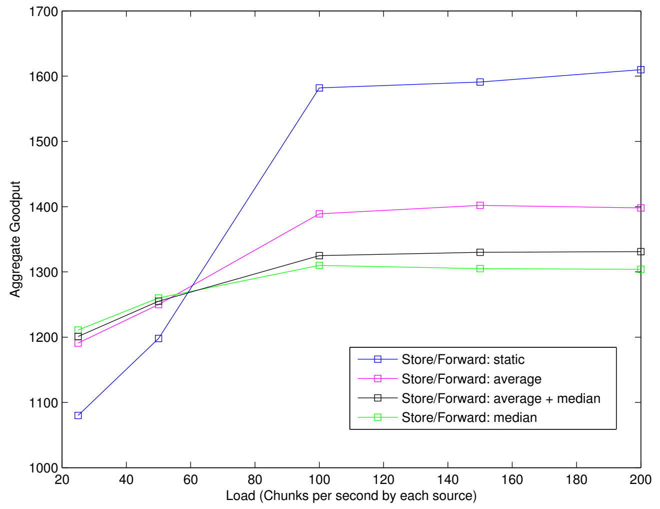


Figure 5.14: Dynamic Store or Forward Decision Threshold: Simulation Results for Hybrid network with wireless destinations (on-off model)

Chapter 6

Future Work

The following aspects of GSTAR protocol needs to be explored further:

- Inter-partition Graph: We have implemented a single copy DTN routing mechanism. This needs to be compared with multiple copy DTN routing mechanisms. Also, GSTAR needs to compared with some existing DTN Routing protocols.
- Finite Storage for Hop-by-hop transport: The current implementation of hop-by-hop transport considers infinite storage. We need to study the effect of various amounts of finite storage along with the associated congestion control on the throughput of the network.
- Storage Aware Routing Metric: The forwarding paths are calculated using SETT as link weights. This path selection metric should be modified to include SETT, LETT and storage available at each node in the route to the destination.

References

[1] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. *ITU T Kaleidoscope: Innov. in NGN*, 2008.

[2] NSF FIA project. `http://www.nets-fia.net`.

[3] Networking technology and systems: Future internet design (FIND), NSF program solicitation, 2007.

[4] Global environment for networking innovations (GENI), NSF program solicitation. `http://www.geni.net`, 2006.

[5] FP7 information and communication technologies: Pervasive and trusted network and service infrastructures, european commission, 2007.

[6] M. Lemke. Position statement: FIRE, NSF/OECD workshop on social and economic factors shaping the future of the internet, January 2007.

[7] New generation networks. `http://www2.nict.go.jp/w/w100/index-e`.

[8] MobilityFirst. `http://mobilityfirst.winlab.rutgers.edu/`.

[9] Samuel C. Nelson, Gautam Bhanage, and Dipankar Raychaudhuri. GSTAR: Generalized storage-aware routing for mobilityfirst in the future mobile internet. In *Proceedings of MobiArch*, 2011.

[10] G. Bhanage, A. Chanda, J. Li, and D. Raychaudhuri. Storage aware routing protocol for the mobilityfirst network architecture. In *European Wireless, DTN Session*, 2011.

[11] J. Burgess, Br. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[12] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. ACM SIGCOMM*, 2007.

[13] Ram Ramanathan, Richard Hansen, Prithwish Basu, Regina Rosales-Hain, and Rajesh Krishnan. Prioritized epidemic routing for opportunistic networks. In *MobiOpp 07*, 2007.

[14] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: a new paradigm for wireless transport. In *Proc. of NSDI*, 2009.

[15] Jörg Ott, Dirk Kutscher, and Christoph Dwertmann. Integrating DTN and MANET routing. In *Proc. of ACM CHANTS*, 2006.

[16] G. F. Aggradi, F. Esposito, and I. Matta. Supporting predicate routing in dtn over manet. In *Proc. of ACM CHANTS*, 2008.

[17] J. Whitbeck and V. Conan. HYMAD: Hybrid DTN-MANET routing for dense and highly dynamic wireless networks. *Comp. Commun.*, 33, August 2010.

[18] S. Gopinath, S. Jain, S. Makharia, and D. Raychaudhuri. An experimental study of the cache-and-forward network architecture in multi-hop wireless scenarios. In *Proc. of LANMAN*, 2010.

[19] Vishnu Navda, Anand Prabhu Subramanian, Kannan Dhanasekaran, Andreas Timm-Giel, and Samir R. Das. CRAWDAD data set sunysb/mobisteer (v. 2007-06-30). Downloaded from http://crawdad.cs.dartmouth.edu/sunysb/mobisteer, June 2007.