MF-TCP : Design and Evaluation of TCP for Message Ferry Delay Tolerant Networks*

Dheeraj Kandula, Parth H. Pathak and Rudra Dutta
Department of Computer Science,
North Carolina State University,
Raleigh, NC, USA.
Email: {dkandul,phpathak}@ncsu.edu, dutta@csc.ncsu.edu

***Abstract*—In message ferry delay tolerant network, certain dedicated mobile nodes (message ferries) facilitate data transfer between disconnected regions of network. Message ferries travel around partitions of network, collecting data from nodes and delivering it to others. Due to high mobility of nodes and network partitioning, such a data transfer can be highly unreliable and it becomes really challenging to meet transport layer communication objectives. As we show, standard TCP performs poorly in such network because of frequent disconnections and longer propagation delays. In this paper, we present a variant of TCP (called MF-TCP) for message ferry delay tolerant networks which is designed to deal with high communication delays and significantly longer periods of disconnections. MF-TCP performs well even when intermediate nodes of TCP connections are highly mobile and round trip latency is unpredictable. Simulation results of MF-TCP show performance improvements over standard TCP in message ferry delay tolerant networks.**

I. Introduction

Delay Tolerant Networks (DTN) are special category of Mobile Ad-hoc Networks (MANETs) where only intermittent network connectivity between nodes is available and network partition may last for significant period of time [1]. Such networks have been proven useful in mission-specific short-term critical applications like battlefields, disaster management etc. Routing in such partitioned sparse networks have been studied rigorously in last few years. Especially, controlled mobility assisted routing has attracted many researchers where certain controlled mobile nodes provide connectivity between disconnected partitions of a DTN. These nodes are often referred as Message Ferries [2], [3], [4].

Message ferry DTN (MF-DTN) is a network where nodes have intermittent connectivity and mobile message ferries facilitates data transfer between disconnected nodes. Such mechanical backhaul where cars or buses (message ferries) carry data between disconnected rural infrastructure (DTN) is in active investigation to provide low-cost Internet access [5]. Researchers have also studied design of optimal ferry route on which ferry can traverse collecting data from some nodes and delivering it to others. Because ferry transit time can be very high, message ferry model has higher propagation delay and queuing delay when compared to other normal connected networks. Most of the research in MF-DTN deals with ferry route design, routing, mechanical backhaul design or power conserving mechanisms. Only few approaches address the problem of actual protocol design for MF-DTN. In this work, we focus on deriving a variant of TCP (MF-TCP) which works efficiently in MF-DTN.

Various TCP variants are proposed for long round trip delays and disconnection conditions in networks. Majority of these variants assume either very short time disconnectivity (as in hand-off) or assistance of other layer protocols to deal with it. In this work, we propose a TCP solution for MF-DTN which works efficiently in long disconnection periods (ferry transit time) and still achieves all transport layer objectives efficiently.

The rest of the paper is organized as follows. Section II discusses unique characteristics of MF-DTN applications and how they affect TCP design. It also points out limitations of various other related TCP variants which does not work efficiently with MF-DTN. Section III elaborates on design and operations of MF-TCP. Performance evaluation and simulation results are presented in Section IV followed by Section V where we conclude and discuss future work.

II. Related Work and Problem Description

We consider single-ferry single-route message ferry system in which single ferry travels on predetermined route at a constant speed. All segments of route where ferry comes in contact with node are mutually non-overlapping. The nodes are stationary or move in relatively smaller predetermined area. It is also assumed that ferry has no restriction on its battery life.

A. Preliminaries

We present a classification of various networking applications based on Round Trip Latency (RTL), throughput and reliability. It will then become obvious that DTN can support only a subset of these applications due to their inherent characteristics.

RTL composes of several delays incurred by different network operations. In a single-ferry single-route DTN, propagation delay (ρ) of a message from source node i to destination j can be given as $\rho = \frac{l_{ij}}{f}$, where l_{ij} is the distance between nodes i and j on ferry route and f is constant speed of ferry. When multiple-ferries multiple-routes are used where ferries synchronize to meet each other at specific contact points on their routes, propagation delay of a message would be summation of message's propagation delay on each individual route. If the ferries interact using throw-boxes, the delay is increased by waiting time of message at each throw-box that

*This work is supported by the U.S. Army Research Office under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI). The contents of this paper do not necessarily reflect the position or the policies of the U.S. Government.

it goes through. Such a propagation delay is similar to one in Ethernet or Token Ring networks but speed of the medium (ferry speed) is far more than actual wired medium.

Transmission delay of message is independent of number of ferries and routes since it only depends on interaction of node and ferry. If M is the size of the standard message and r bits/sec is the rate of the interface, transmission delay (τ) can be given by $\tau = \frac{M}{r}$. Queuing delay of the message at a node is the time from when the message is generated and queued for transmission at node to its delivery to ferry. In the case of static nodes, ferry visits the node in every trip. Here, worst case queuing delay (q) of the message can be given by $q = \frac{L}{f}$ where L is total distance traveled by ferry in single trip (or length of route). Note that mobile nodes scenario is handled differently because such nodes might not always be in range of ferry when ferry is nearby during its trip. Buffer overflow or trip timeout - two events can force a mobile node to move towards the ferry route to deliver its messages. If the node has generated enough messages to be delivered to ferry before it exceeds its buffer capacity, it must move towards ferry route. Even if the node's message generation rate is less, it should frequently visit the ferry to avoid infinite delays for few generated messages and also for messages to be received by the node from the ferry. Such trip timeout are often set to multiple (k) of ferry route time to trigger the node movement on every k^{th} trip of ferry. Queuing delay in both cases would be $\frac{L}{f} < q \leq \left(k \cdot \frac{L}{f}\right)$. Also, medium access time before the node can send data to ferry is negligible due to lesser contention chances in sparse DTN. Thus, the Round Trip Latency of a message from source to destination in DTN can be given by $RTL = \rho + \tau + q$.

Throughput of every node in single-ferry single-route $n-$node DTN can be given by $\frac{r}{2n}$ bits/sec. With assumptions that ferry route is optimized for no idle periods without data transfer, node-ferry contact times are identical for all nodes and ferry and node have equal amount of data to be sent to each other in every trip, a node can send $\frac{Lr}{2nf}$ amount of data to ferry every trip. Thus, a node can achieve an aggregate throughput of $\frac{r}{2n}$ bits/sec per trip in best-case stable system over time.

Reliability in message ferry DTN is significantly affected by high RTL. Majority of the transport protocols rely on RTL to estimate the message losses. Wireless data transfer between ferry and node can also be a reason of such data loss. Hence, the transport protocol should be able to distinguish between different types of losses and should adapt to high latency.

Above analysis shows that applications which can tolerate high RTL and low throughput (e.g. e-mail) are only suitable in message ferry DTN. If the buffer capacity of the ferry and even nodes is large, high throughput applications (e.g. file transfer) can also be supported. These characteristics are the basis of transport protocol design for MF-DTN which we discuss next.

B. Transport Protocol Requirements

Transport protocols like TCP or SCTP provide end-to-end services for data transfer such as in-order delivery, high reliability etc. When well-known transport protocols are ported to MF-DTN, their performance is either degraded or they completely fail to perform due to high delay and disconnections. Protocols like TCP depend on timers for detection of losses which in turn are estimated based on RTL. Naively setting timers to high value does not always solve the problem because actual time taken by ferry in transit can change frequently based on various factors. Hence, connection establishment-termination, reliable data transfer and estimation of RTL should be improvised with different methodology. Also, buffer constraints should also be properly enforced and separate connection admission control algorithm should be implemented to avoid congestion and buffer overflow problems.

TCP design of MF-DTN can be compared with mobile nodes in Wireless LANs where clients move from one access point to another. In MF-DTN, mobile ferry moves from the coverage area of all nodes. Such hand-off requires proper handling of continuing TCP connections. Though losses in both MF-DTN and WLAN can be due to node mobility, wireless medium or congestion, disconnection time can be much higher in MF-DTN. Various TCP variants are proposed for WLAN hand-off scenarios but we show below that all such protocols either perform inefficiently or completely fail in MF-DTN.

In Freeze-TCP [6], the receiver is expected to send a Zero Window Advertisement to sender at least one RTT prior to disconnection so that the sender can stop transmitting any more data immediately. This forces the other end of connection to go to persist mode and prevents unnecessary slow-start upon reconnection. Freeze-TCP fails to work in MF-DTN since the disconnection is due to mobility of intermediate node (ferry), not the sender or receiver. The ferry neither responds to sender's SYN nor can deliver it to receiver before connection times out. Even if the timers are set to high value to reflect ferry transit time, throughput will be very less because the ACKs don't arrive until the ferry comes back to sender.

In Snoop-TCP [7], foreign agent before wireless last hop buffers packets until it is correctly received by mobile host. This prevents unnecessary end-to-end retransmissions. It is mainly designed where last hop is assumed to be lossy due to wireless medium or mobility. In MF-DTN, it is not advantageous even if the ferry buffers data since end-to-end connection would still time out when ferry moves out of node's range. Also, ferry does not acknowledge any data which reduces throughput even with high timer values.

In Multiple TCP (MTCP [8]), the connection from mobile host to remote fixed host is split into two separate TCP connections at the base station - one from mobile host to base station and other from base station to remote fixed host. In MF-DTN with MTCP, node establishes connection with ferry and transfers data. When ferry moves out of the range, node's connection times out and is aborted. After reaching to destination node, ferry establishes a connection and transfers data. Similar time out will occur at destination once ferry is out of range. Note the if the time out is set to high value, MTCP results into relatively higher throughput even in MF-DTN. The only limitation of MTCP is that connections that are initiated when ferry is out of range can not be handled and would result into unnecessary retries. In battery-

operated limited-power mobile nodes, this should be avoided intelligently. Similar problems occurs with I-TCP [9]. M-TCP (Mobile TCP [10]) uses a split TCP approach as in MTCP but the ACKs are not generated by the device where the split occurs, instead the original ACKs are forwarded to the required destination. M-TCP does not work well in MF-DTN because ferry does not acknowledge SYN segment from source node. Connections initiated when ferry is out of range are also aborted after several retries. TCP Spoofing [11] can be used in MF-DTN where intermediate node acknowledges data on behalf of destination. If the ferry does so, it can initiate connection with sender node and can perform data transfer until it is in range. After this the connection is aborted due to ferry mobility and none of the segments are delivered to destination.

It becomes clear that most of the related TCP variants don't fit the MF-DTN because they either do not consider highly mobile intermediate nodes (ferry) or longer disconnection times. Hence, we are interested in building a TCP variant that will act as a convergence layer for applications (with certain types of data transfer requirements) on a MF-DTN. In the next section, we present Message Ferry TCP (MF-TCP), a variant of TCP for message ferry delay tolerant network.

III. MF-TCP : Message Ferry TCP

First, we summarize the suggested modifications in standard TCP and then present the details of MF-TCP.

A. System Design

MF-TCP design is based on two fundamental requirements. First, ferry should acknowledge to sender as an intermediate node because connection is either not established or aborted mostly due to its mobility. Second, instead of setting the TCP timer to high values, the timers should be started only during node-ferry contact time and should be stopped afterwards. This is feasible in networks like MF-DTN because the ferry mobility is controlled and known in priori.

In order to ensure that TCP works in MF-DTN, several changes are made to TCP. MF-TCP at the nodes stores application data when ferry is not in communication range. The timers are started on node or ferry only when they are in communication range of each other. Ferry maintains a list of connections to all nodes and estimates amount of data that will be sent on every connection. When ferry is in communication range of a node, it sends a window update for every connection on the ferry. This opens up the windows on the node and data transfer begins at high speed. The windows are closed only when ferry moves out of range. The nodes also estimate the amount of data that will be sent to the ferry as it stores all the data before ferry is in range. At the node, if there is a new request from an application, the node satisfies the request only if there is sufficient bandwidth to fulfill the request. Else the node stores this request and fulfills it during the next trip of the ferry. This way, congestion is explicitly avoided by not allowing new connections if the ferry does not have enough buffers to hold data. When the ferry leaves the node, it sends a Zero Window Update for every connection on the node so that the TCP connections on the node enter persist state.

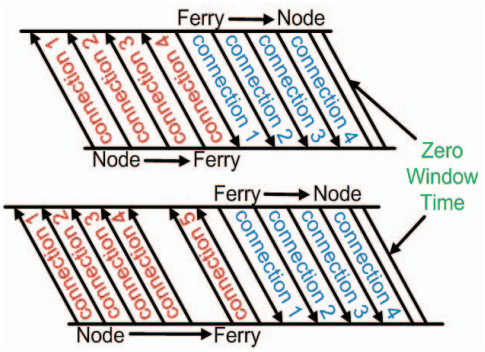

Fig. 1: Timing Diagram - entire bandwidth occupied by connections (upper), unused bandwidth occupied by new connection (lower)

1) Window Size Estimation - Node: The node stores the data received from the application in its internal buffer specific for every TCP connection. The duration when the ferry is in communication range only accounts for calculating communication window of node. Upon ferry's contact, MF-TCP on the node estimates the amount of data that can be sent for each connection. This estimation is based on the number of connections on the node and the window size advertised by the ferry (See Fig. 1). The window size at the node for every connection $Connection_{WindowSize}$ can given by –

$$Connection_{WindowSize} = \frac{T * B}{Number\ of\ connections + 1} \quad (1)$$

Where T is time of meeting time of node and ferry and B is wireless channel bandwidth.

From the window size per connection, the amount of data that can be sent to the ferry $Data^t_{ferry}$ (without considering ferry's acknowledgments) is calculated as –

$$Data^t_{ferry} = Min(Connection_{WindowSize}, Ferry_{WindowSize}) \quad (2)$$

The number of connections is incremented by 1 to allow any new connection that is requested after the estimation. Thus, the maximum amount of data that can be transferred between a node and the ferry is limited by the number of connections from the node and the ferry's advertised window size. However, after the estimation if a new connection request arrives, it is satisfied only if there is sufficient bandwidth left. When the amount of data transferred by every connection is less, most of the window can be wasted. This excess window bandwidth is allocated to new requests. This is shown is Fig. 1. However, if the data transferred from the node is significantly large, only one extra connection will be allowed.

Once the window size has been estimated, the bandwidth for acknowledgments has to be allocated. This is done to ensure that the acknowledgments are received from the ferry. The previously estmated window size is decremented by the maximum expected number of acknowledgments. Number of ACKs equals number of segments sent by node to ferry. The equations below evaluates the final value for window size –

$$ACK_{count} = \frac{Data^t_{ferry}}{MTUSize} \quad (3)$$

$$Data_{ferry} = Data^{t}_{ferry} - (ACK_{count} * ACKSize) \quad (4)$$

2) Window Size Estimation - Ferry: Ferry on arrival at a node determines the number of connections destined for that node. The ferry also calculates the window size allowed for every connection and advertises it to the node. A similar set of equations are used to estimate the window size for every connection before sending any data to the node. The window advertise from ferry opens up window for every connection on the node. $Connection_{WindowSize}$ for ferry is equivalent to Equation (1). From the window size per connection, the amount of data that can be sent to the node $Data^{t}_{node}$ (without ACKs) can be calculated as -

$$Data^{t}_{node} = Min(Connection_{WindowSize}, Node_{WindowSize}) \quad (5)$$

Data may already exist in the ferry's buffer as leftover from its previous trips. This amount of data (B) is deduced from the calculation of available window size on the ferry as below–

$$Data^{t}_{node} = Data^{t}_{node} - B \quad (6)$$

$$ACK_{count} = \frac{Data^{t}_{node}}{MTUSize} \quad (7)$$

$$Data_{node} = Data^{t}_{node} - (ACK_{count} * ACKSize) \quad (8)$$

B. TCP State Transitions

The state transition diagram of MF-TCP at the node does not change. It is the same as the standard TCP mentioned in RFC 793 [12]. The state transition diagram at the ferry is shown in Fig. 2. When a node initiates a new connection with the ferry, the connection on the ferry progresses through a set of states. The various states are : INIT, ESTABLISHED-1, ESTABLISHED-2, ESTABLISHED, FIN-1, FIN-ACK, FIN-2, CLOSING and CLOSED. CLOSED state is imaginary as the connection is either not established or terminated.

INIT : When a SYN is received, the connection enters this state. In this state, MF-TCP initializes the internal buffers and sends a SYN ACK for the SYN received. Once the SYN ACK has been sent, the connection moves to ESTABLISHED-1 state.

ESTABLISHED-1 : In this state, the connection sends ACKs for segments received from the node and stores the segments in the internal buffer. The ESTABLISHED state is not entered yet since the SYN from the other end of the connection is not received. SYN from the other end will be received only when the ferry reaches the destination. If the connection receives a FIN segment from the node, the connection moves to ESTABLISHED-2 state.

ESTABLISHED-2 : The connection stays in this state until it receives the SYN from the other end of the connection.

ESTABLISHED : The connection in this state performs normal data transfer as in standard TCP. The connection ACKs segments received from either end of the connection and stores only the data segments in its internal buffer. The ACKs from receiver destined to sender are suppressed.

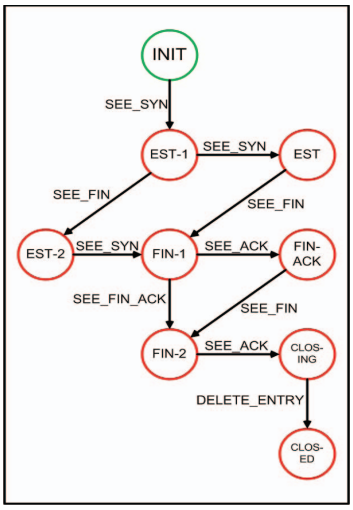

Fig. 2: Ferry - TCP State Transition Diagram

FIN-1 : The connection enters this state when a FIN segment has been received from one end of the connection. The connection ACKs this segment and stores the FIN segment in the internal buffer.

FIN-ACK : The connection enters this state when the actual ACK for the FIN from the other end is received. This helps the connection to keep track of the actual ACKs received.

FIN-2 : The connection enters this state when the second FIN is received. This FIN has to be from the other end of the connection.

CLOSING : The connection enters this state when the ACK for the second FIN is received.

CLOSED: The connection enters this state when the four way connection termination is complete. This state represents the removal of the connection or the absence of a connection.

C. TCP Operations

We take an example of a TCP connection initiated by node 2 to node 4 (See Fig. 3.) to describe following operations of MF-TCP.

1) Session Initiation: Session initiation at a node is similar to standard TCP except that the ACKs in the three-way initiation are pseudo ACKs. When the ferry is within communication range, node 2 initiates a connection by sending a SYN segment to node 4 and starts a retransmission timer. The connection at node 2 moves to SYN-SENT state. The ferry intercepts the SYN segment and responds with a SYN ACK segment including MTU and window size. The ferry stores the SYN in its internal buffer and starts a retransmission timer for the SYN ACK and progresses the connection to ESTABLISHED-1 state. When node 2 receives the SYN ACK segment from the ferry, an ACK is sent back and the retransmission timer is stopped. The node's connection moves to ESTABLISHED state.

If any of the segments are lost, the retransmission timer expires and the segments are retransmitted utmost 3 times. The connection is aborted after that by transmitting a RESET segment.

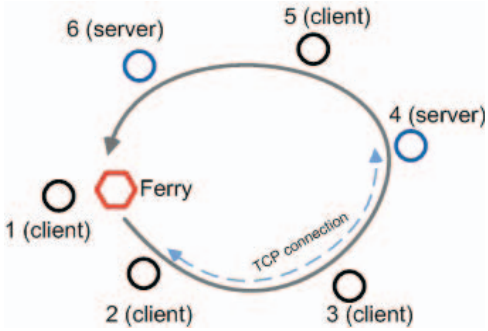

Fig. 3: MF-TCP: Experimental Network Topology

2) Data Transfer: Data transfer between the node and the ferry is significantly different from standard TCP. When the node's connection is in ESTABLISHED state, the node sends segments to the ferry according to the window size estimated by Equation (4). instead of initiating slow start phase as in standard TCP. If the amount of data available at the node is less than the window size, the window remains open for any new data transfer. But if the node's data exceeds the window size, the node transfers data only on the next trip of the ferry.

Node 2 transfers data according to its permitted window size to the ferry and starts a retransmission timer. The ferry ACKs each and every segment. The ferry stores the segments in its internal buffers to later transfer them to node 4. However, if there is any segment loss, the retransmission timer times out and the segment is retransmitted only if the window is open. If the window closes, the node enters persist state and no retransmissions occur. However, the node does not send zero window probes as the bandwidth allocated for this node has been already consumed. Before the ferry goes out of communication range of node 2, the ferry transmits a zero window update for the connection. The zero window update is transmitted for each and every connection on the node. The zero window is sent significantly ahead of time to avoid out of communication range losses. One advantage of MF-TCP is that even if zero window updates are lost, the node does not send any more data beyond the permitted limit. This is because the node will anyways close down its window when the transmission duration elapses.

When the ferry reaches node 4, the ferry transmits segments according to the window size of node 4 and starts a retransmission timer. When node 4 updates it's window size, the ferry transmits further segments until the transmission duration elapses. The ACKs received by the node are not stored in the ferry as the ferry locally ACKs the segments instead of end-to-end ACKs. If there are any segment losses, the retransmission timer times out and the segments are resent based on the window size and the remaining bandwidth allocated to the connection. This way, the bandwidth available between the ferry and the node are utilized to the maximum extent possible. Note that in MF-DTN, segment losses occur mainly due to buffer overflow at the various layers since chances of MAC layer collisions are less in sparse DTN.

3) Session Termination: Session termination at a node is similar to standard TCP except that the ACKs in the four-way termination are pseudo ACKs. When node 2 sends a FIN segment for the connection, the ferry intercepts the FIN segment and responds with a pseudo ACK on behalf of the destination and stores the FIN segment. The ferry's connection moves to FIN-1. The reception of the ACK at the node moves the connection to FIN-WAIT2.

When the ferry reaches node 4, it transmits the FIN segment on behalf of node 2 and starts a retransmission timer. On reception of the FIN, the node's connection state changes to CLOSE-WAIT state. When the ferry receives the actual ACK from the node, the connection's state changes to FIN-ACK. When data transfer from node 4 is complete, node 4 transmits a FIN segment to ferry and moves to LAST-ACK state. Ferry's connection responds with an ACK and connection state changes to CLOSING. Node 4 moves to CLOSED state on reception of the ACK and the connection is cleared.

If any of the segments are lost, the segments are retransmitted thrice and if the limit is exceeded, the connection is aborted by sending a RESET segment. However if the transmission duration ends before the ACK is received, the node's connection enters persist state on the reception of a zero window update from the ferry. When the ferry revisits node 4, the window opens up and the connection resumes.

IV. Performance Evaluation

A. Simulation Design

We model the proposed message ferry delay tolerant network and MF-TCP in OPNET. The simulation topology (See Fig. 3.) consists of 7 nodes – 4 of which take up the role of clients, 2 of them act as servers and a special node called the ferry. All the nodes are equipped with an 802.11 wireless interface and omni-directional antenna. The client and server nodes are static and the ferry route passes through the coverage area of all nodes. For simulation purposes, we assume that ferry route is designed in such a way that ferry is work-conserving and does not roam idly. Node 2 initiates a TCP connection with node 4 by sending an FTP request.

In the simulation, the communication window is equally shared by the ferry and the node. The time difference between ferry arrival and departure is used to estimate the communication window size. The ferry on arrival near a node estimates the amount of data for every connection of node and advertises this information to it as window update. For fairness, ferry splits apart its communication window across all connections irrespective of the amount of data available for each connection. The ferry transfers data to the node either until the node window size becomes zero or its communication window timer expires. Now, ferry receives segments from the node and inserts them into a per-connection list ordered by sequence number of segments. In MF-TCP, node transmits any available data until either it runs out of ferry window size or the departure of the ferry, whichever occurs first.

B. Numerical Results

Simulations for MF-TCP were performed in which a 10 KB file is transferred from server (node 4) to client (node 2).

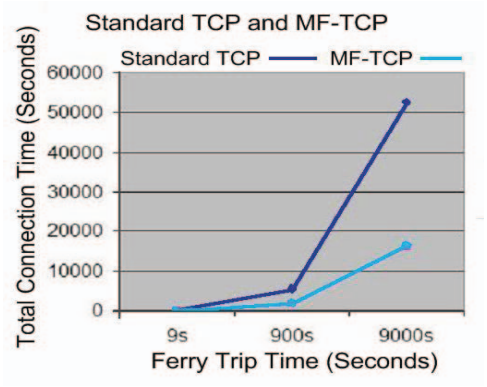

Fig. 4: Throughput comparison of standard TCP and MF-TCP

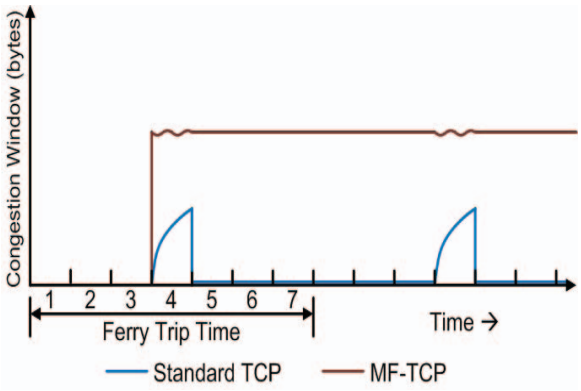

Fig. 5: Comparison of congestion Window Size for receiver node in standard TCP and MF-TCP

The ferry trip causes disconnection in ongoing data transfer and hence we use ferry trip time as a comparison metric for total connection time in transferring 10 KB file. As the ferry trip time increases, total time required for file transfer also increases drastically in standard TCP (See Fig. 4.). While MF-TCP utilizes the ferry buffers more efficiently since server can send many more segments to ferry in every trip. It was observed that with reasonable buffer size at ferry, smaller file transfer can be completed in as low as one trip. Transferring ACK for final FIN segment from one side to another may require an extra trip from ferry before connection can be terminated gracefully. It is worth noting that even with more TCP connections, the performance remains the same since there is no explicit congestion with MF-TCP. The number of connections supported by the network is restricted by the size of buffer which turns out be realistic in MF-DTN. Note that it is not possible to compare the MF-TCP performance with other TCP variants described in Section II-B because they simply do not work in the scenarios of interest. Fig. 5. shows the approximate behavior of congestion window size of server for standard and MF-TCP. In standard TCP, node 4 (server) realizes the presence of ferry, it starts with slow-start phase and keeps on increasing congestion window. When ferry moves out of its range, it senses disconnection and the congestion window drop to one segment. After many retries without acknowledgment, the connection is aborted. Similar scenario is repeated in every trip when ferry enters and leaves its coverage area. While in MF-TCP, when ferry enters the coverage area of server, it advertises its window size and data transfer begins at a higher speed without slow-start phase. When ferry is about to move out of server's coverage area, it sends a zero window advertisement to server node. This freezes the congestion window of the server (straight horizontal line in Fig. 5.) node which resumes operations only when the ferry is again in range during its next visit. Retransmission Time Out (RTO) slowly converges to ferry trip time in standard TCP but remains very low and consistent in MF-TCP. This is due to the fact that timers in MF-TCP are started only when node and ferry are in contact of each other.

V. Conclusion and Future Work

We have described design and operations of MF-TCP, a variant of TCP for message ferry delay tolerant networks. MF-TCP utilizes pre-calculated window size and pseudo ACKs for data transfer which results into efficient utilization of ferry buffers even in case of frequent disconnections. In future, we plan to evaluate MF-TCP with more simulations involving more general DTN scenarios. This will also include evaluation in larger DTN with random topology and traffic pattern. Fairness study of connection admission control and its relation with ferry buffer size and management would also be an interesting direction for future work. Design and analysis of MF-TCP for multiple ferries, multiple routes and sharing techniques like throw-boxes is part of our ongoing research.

References

[1] DTN Research Group (DTNRG). [Online]. Available: www.dtnrg.org
[2] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2004.
[3] W. Zhao and M. Ammar, "Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks," *Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of*, May 2003.
[4] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, March 2005.
[5] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav, "Low-cost communication for rural internet kiosks using mechanical backhaul," in *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*. New York, USA: ACM, 2006.
[6] T. Goff, J. Moronski, D. Phatak, and V. Gupta, "Freeze-tcp: a true end-to-end tcp enhancement mechanism for mobile environments," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, Mar 2000.
[7] E. Amir, H. Balakrishnan, S. Seshan, and R. Katz, "Efficient tcp over networks with wireless links," *Hot Topics in Operating Systems, 1995. (HotOS-V), Proceedings., Fifth Workshop on*, pp. 35–40, May 1995.
[8] R. Yavatkar and N. Bhagawat, "Improving end-to-end performance of tcp over mobile internetworks," *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, Dec. 1994.
[9] A. Bakre and B. Badrinath, "I-tcp: indirect tcp for mobile hosts," *Distributed Computing Systems, 1995., Proceedings of the 15th International Conference on*, pp. 136–143, May-2 Jun 1995.
[10] K. Brown and S. Singh, "M-tcp: Tcp for mobile cellular networks," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 5, pp. 19–43, 1997.
[11] J. Ishac and M. Allman, "On the performance of tcp spoofing in satellite networks," *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, pp. 700–704 vol.1, 2001.
[12] "Transmission Control Protocol Specification," RFC 793, 1981.