On the Optimization and Comparative Evaluation of a Reliable and Efficient Caching-Based WSN Transport Protocol

Nestor Michael C. Tiglao†, António M. Grilo∗

∗INESC-ID/INOV/IST, Rua Alves Redol, N° 9, 1000-029 Lisboa, Portugal
†EEEI, University of the Philippines, Diliman, Quezon City, 1101 Philippines
Email: nestor.tiglao@inesc-id.pt, antonio.grilo@inov.pt

Abstract—Wireless Sensor Networks (WSNs) have been envisioned for mission-critical applications such as critical infrastructure protection where reliable data delivery, goodput, and energy efficiency are of paramount importance. Due to the resource constraints in WSN devices, several transport protocols such as RMST, PSFQ, DTC and DTSN have been designed to leverage intermediate caching in order to avoid the costly end-to-end retransmissions inherent to traditional transport protocols such as TCP. Transmission window size adaptation, acknowledgment semantics, and loss recovery are important components in the design of the transport protocols. TCP uses the additive-increase multiplicative-decrease (AIMD) algorithm and cumulative ACK mechanism. More recent works such as DTPA have shown that TCP’s AIMD scheme leads to inefficient performance in wireless networks. DTPA uses a fixed-sized transmission window based on the bandwidth-delay product (BDP) of the path but retains the end-to-end semantics. However, with caching based protocols, there is a need to revisit the transmission window size optimization, since the latter has a strong impact on the effectiveness of the cache. In this paper, we provide two optimization schemes namely, an enhanced O(1)-time complexity NACK-based repair mechanism and the optimal transmission window for DTSN. Incorporating these optimizations, we implemented an enhanced DTSN protocol (denoted as DTSN+) and compared its performance with TCP and DTPA. We show that the optimal transmission window of DTSN+ is dependent on the average cache size at the intermediate nodes. Our results show that DTSN+ will in general significantly outperform both TCP and DTPA in terms of goodput and energy efficiency.

Index Terms—reliability; transport; adaptive MAC; intermediate caching; WSN; repair

I. INTRODUCTION

A Wireless Sensor Network (WSN) is typically composed of small autonomous resource-constrained devices that transmit data from sensor nodes to one or more sink nodes. Since WSN nodes operate autonomously, power saving techniques are usually complemented with low power radio communications that lead to multi-hop data transmission from the sensor nodes to the sink nodes and vice versa. Besides leading to a lower throughput, multi-hop communications are also the cause of additional interference and hidden terminal problems due to spatial reuse, which, complemented with the fact that most WSN standards such as IEEE 802.15.4 specify operation in unlicensed ISM bands, means that radio links are usually more error-prone than in typical WLANs.

The traditional design of reliable transport protocols has been end-to-end delivery of data segments from the source to the destination transparent to the intermediate nodes along the path (e.g., TCP). However this semantic leads to poor performance (in terms of throughput and energy efficiency) when applied to WSNs. Thus, new semantics have been developed where part of the transport layer’s function is packet loss detection and recovery of lost segments, which can be performed either end-to-end or hop-by-hop. In the end-to-end semantic, the end-points (sender or receiver) are responsible for loss detection and initiating loss recovery. On the other hand, in the hop-by-hop semantic, the intermediate nodes along the path from the sender to the receiver are responsible for these functions.

WSN protocols can be classified into two general classes based on how reliability is guaranteed or achieved, namely (1) packet-based and (2) event-based [1]. With packet-based reliability (e.g., RMST [2], PSFQ [3], DTSN [4]), lost packets are detected at the sink and/or at intermediate nodes and a retransmission scheme is used. Loss signalling uses some kind of acknowledgment mechanism and loss recovery is performed either in a hop-by-hop or end-to-end manner. Furthermore, protocols that implement packet-based reliability can leverage on intermediate caching where cache points are able to help in retransmitting lost packets. On the other hand, event-based reliability (e.g., ESRT [5]) guarantees reliability through end-to-end source rate adjustment, where the perceived reliability at the sink is signalled back to the source. The source increases the rate if the current reliability is not met (subject to congestion constraints) and reduces the rate if the required reliability is exceeded.

Transmission window adaptation and acknowledgment semantics play crucial roles in the performance of transport protocols. TCP performs end-to-end delivery and loss recovery. It uses the additive-increase multiplicative-decrease (AIMD) scheme to adapt the transmission window as part of it congestion control function. However, such AIMD scheme has been shown to lead to TCP’s inefficiency in wireless ad hoc
networks [6]. DTPA uses a fixed transmission window based on the bandwidth-delay (BDP) product of the path in a wireless network, i.e., BDP+3. The caching function employed in many WSN transport protocols necessitates a study in the optimal transmission window since the BDP obtained in previous studies may no longer hold for such class of protocols.

**Paper contributions:** This paper makes the following contributions: (1) We have developed an enhanced NACK repair mechanism with O(1) algorithmic complexity; (2) We have studied the optimal transmission window for DTSN; and (3) We have implemented the DTPA [6] protocol and have analyzed its performance together with TCP and our enhanced DTSN protocol under high frame error rates. We performed extensive simulations to analyze the performance improvement of the enhanced DTSN in terms of goodput and energy efficiency.

The rest of the paper is organized as follows. Section II presents the related work. In Section III we discuss the optimization and enhanced mechanism. In Section IV we discuss the simulation environment. We present our results in Section V. Finally, section VI concludes the paper.

II. RELATED WORK

In this section, we present the WSN protocols that are most closely related to our work.

Reliable Multi-Segment Transport (RMST) [2] offers two simple services: data segmentation/reassembly and guaranteed delivery using a selective NACK-based ARQ mechanism. RMST is designed to run on Directed Diffusion routing protocol and can be configured for in-network caching and repair. It performs hop-by-hop loss recovery by detecting lost packets and sending a timer-driven NACK to the previous node. However, the repair timeout must be set appropriately because setting it too long will give low good put while setting it too short will lead to congestion. In contrast, our enhanced NACK repair mechanism does not require any timers and has an O(1) time complexity making it suitable for implementation in WSN nodes.

Pump Slowly Fetch Quickly (PSFQ) [3] is a protocol primarily designed for downstream multicast dynamic code update, though it can also be configured for unicast communication. Regarding intermediate caching, data is reconstructed at each hop. While this makes sense for dynamic code update (i.e., each node must get all the executable code fragments), that can be very limiting for other applications (audio or image transmission) since it poses significant requirements on node storage capabilities. Furthermore, PSFQ is a rather hop-by-hop reliability protocol than an end-to-end solution. Hence, end-to-end reliability cannot be maintained in all cases.

Tunable Reliability with Congestion Control for Information Transport (TRCCIT) [8] provides probabilistically guaranteed tunable reliability using localized techniques such as probabilistic adaptive retransmissions, hybrid acknowledgment and retransmission timer management. TRCCIT proactively alleviates network congestion by opportunistically transporting the information on multiple paths. TRCCIT fulfills application reliability requirements in a localized way by adapting the MAC retry limit so as to maintain end-to-end application reliability. In contrast, our approach adapts the MAC retry limit by guaranteeing per-node MAC-layer reliability and only requires local knowledge of link quality (e.g., LQI).

Distributed TCP Caching (DTC) [7] enhances TCP in order to make it more efficient in WSNs. DTC improves the transmission efficiency by compressing the headers and by using cache at selected intermediate nodes. DTC is fully compatible with TCP, leaving the endpoints of communication unchanged it only requires changes in the logic of intermediate nodes. The said papers only considers the case where intermediate nodes can only cache at most one segment. For multiple segments, the authors propose the use the TCP SACK option. DTC, just like TCP, uses the AIMD algorithm for transmission window adjustment. However, this mechanism has been shown to be lead to inefficient performance in wireless networks [6].

Datagram Transport Protocol for Ad Hoc Networks (DTPA) [6] is a datagram-oriented end-to-end reliable transport protocol for ad hoc networks that incorporates two techniques: a fixed-size window-based flow-control algorithm and a cumulative bit-vector-based selective ACK strategy. The size of the transmission window is a function of the wireless bandwidth-delay product (BDP) [12]. In the case of 802.11-based network, BDP is equal to \(\frac{1}{n}\) where \(n\) is the path length in hops. DTPA sets the optimal transmission window equal to \(\frac{1}{n} + 3\). DTSN also uses a cumulative bit-vector acknowledgment strategy employing both NACK and ACK control packets. In this study, we set the DTSN transmission window to a fixed value based on the optimal transmission window we obtained and verified experimentally. In contrast to DTSN, DTPA does not support intermediate caching.

In [9], we developed an analytical model for transport layer caching in wireless sensor networks and proved analytically that intermediate caching significantly improves transport layer protocol performance.

III. DTSN OPTIMIZATION

Distributed Transport for Sensor Network (DTSN) [4] supports both full as well as differentiated reliability and employs selective repeat ARQ to improve energy efficiency. It uses both positive acknowledgment (ACK) and negative acknowledgment (NACK) sent from the receiver upon the request of the sender through an Explicit Acknowledgment Request (EAR) within a EAR retransmission interval. The EAR signal can also be piggybacked on to a data packet. A NACK contains a bitmap of missing packets detected by the receiver. While relaying such NACKs, intermediate nodes will learn about the missing packets and check if those packets are present in their cache. If so, the intermediate nodes will retransmit those packets towards the receiver and modify the NACK bitmap accordingly before sending it towards the sender. Fig. 1 shows an example where packet 2 is lost. Upon receiving the EAR, the receiver (node 3) sends a NACK. The sender (node 0) receives the NACK and retransmits packet 2 (i.e., end-to-end). DTSN only sends ACK/NACK packets in response to an EAR.
However, the basic DTSN specification does not specify the appropriate size of the transmission window, which in the DTSN protocol is called acknowledgment window (AW). Furthermore, the performance of DTSN has never been compared to other transport protocols. This paper addresses both issues as well as proposed an enhanced repair mechanism. We compared DTSN with TCP in order to verify TCP’s inefficiency resulting from the AIMD transmission window adaptation. Although, TCP does not support intermediate caching, DTC maintains the AIMD feature of TCP. We also chose to compare DTSN with DTPA because both of them use a fixed transmission window although DTPA does not support caching. Our goal is to demonstrate the performance gain that can be achieved by using the optimal transmission window (fixed) and intermediate caching.

A. Transmission Window Optimization

The basic DTSN specification does not specify the optimal value of the acknowledgement window (AW). In this study, we experimentally verified and confirmed the optimal AW (AW_opt) for the given network scenarios we considered (as discussed in section IV). In order to simplify our scenarios, we set a fixed cache size (CS) of 20 packets for each intermediate node and varied the transmission window at the DTSN sender.

The goodput performance of DTSN is dependent on the size of the transmission window in relation to the cache size at the node. For AW ≪ CS, DTSN will not be able to maximize the probability of caching in-flight packets leading to low throughput. For AW ≫ CS, the DTSN sender will send too many in-flight packets and the probability of caching all those packets will also decrease as the value (AW-CS) increases thereby also leading to sub-optimal performance. The optimal AW value is within the range [CS, CS + Δ] where Δ is around 10 based on our results.

The explanation for the Δ is due to the cumulative selective acknowledgment and caching feature of DTSN. For example, if at a given instant a packet with sequence number (seqno=N) arrives at the receiver DTSN receiver, it is possible that packets with seqno in the range [N − 2Δ, N + 1Δ] could have been in transit and cached at intermediate nodes. If any of those packets are detected as lost, the DTSN receiver will send a NACK back to the sender, and those packets could then be recovered from the intermediate nodes. We will discuss the simulation results in section V.

B. Enhanced NACK Repair Mechanism

The NACK Repair mechanism is designed to proactively detect and recover lost packets at the intermediate nodes in order to speed up the delivery of those packets at the destination. We present an enhanced version of the NACK repair mechanism that we proposed in [10]. This previous work only considered relatively low throughput (i.e., 50 kbps) constant bit rate traffic with a small transfer size of 20 packets, whereas in this study, we consider stream-type data transfer (i.e., TCP-like). This enhanced mechanism has O(1) time complexity requiring only an out-of-sequence detection operation and repairs only a single loss packet at a time as shown in Algorithm 1.

The NACK Repair mechanism is implemented at the intermediate nodes where several state variables are used, namely:
- next_ – The next sequence number expected.
- maxseen_ – The highest sequence number seen so far.
- repseqno_ – The flag to indicate that an RNACK has been sent to the previous-hop node.
- repseqno_ – The sequence number of the last RNACK (indicating out-of-sequence packet).

Whereas in [10], we clear the repseqno_ flag immediately after sending the RNACK, Here, we clear this flag only upon reception of the lost packet indicated in the repseqno_. It has been discovered that the former approach induces a lot of duplicate RNACK-recovered packets.

Fig. 2 shows an example of how the NACK repair mechanism operates at a given node, say node n, the nth node from the source. In this example, node n receives packets 1 and 2 in order but then receives packet 4. Packet 3 is deemed lost since packet 4 is not equal to the expected sequence number next_=3. Node n then raises the reppending_ flag to 1, stores repseqno_=3 and sends an RNACK packet to node n – 1. Meanwhile, as node n receives more packets, it updates the value of maxseen_ if segno > maxseen_. When node n receives packet 3, it clears repending_ to 0 and sets next_=maxseen_+1. Then, node n receives packet 8, which triggers a new round of the repair mechanism.

C. Adaptive MAC retry limit

In this study, we also incorporate the adaptive MAC retry limit mechanism proposed in [10]. Based on equation 1,
Enhanced NACK Repair Mechanism

procedure PKT_RECV(PKT)
...
if (!rpending_ && seqno != next_)
then
repseqno_ ← seqno
rpending_ ← 1
▷ raise Repair Pending
Send RNACK (seqno)
else
do nothing
end if
if (rpending_ && seqno == repseqno_)
then
rpending_ ← 1
▷ clear Repair Pending
next_ ← maxseen_ + 1
▷ update next_
end if
if (seqno > maxseen_)
then
maxseen_ ← seqno
▷ update maxseen_
end if
...
end procedure

IV. SIMULATION ENVIRONMENT

We implemented the DTSN protocol and the NACK repair mechanism in ns-2.31 [11] and conducted extensive simulations. We consider a linear network topology consisting of 10 nodes with a single source (node 0) and destination (node 9). All the intermediate nodes have a cache size of 20 packets unless specified otherwise. The source sends 500 packets of 500 bytes each in a stream-type transfer similar to TCP. We consider two simulated network scenarios as described below and vary the Frame Error Rate (FER) from 0 to 0.70. The MAC retry limit is set to 3 (default value) unless specified otherwise. The DTSN EAR interval is set to 200 msec which is equal to the default minimum RTO setting in ns-2.

In order to conduct a comparison and analysis of the effect of our mechanisms on protocol performance, we fix the FER and the network topology for each case. For each experiment, we conducted 20 simulation runs and obtained the 95% confidence intervals. Table II provides a summary of the simulation parameters.

- Scenario 1 (Global hotspot scenario): All nodes in the network uniformly experience the same FER as shown in Fig. 3. This scenario can be considered as a worst-case scenario since no nodes in the network have lower FER and all transmitted packets are subjected to the same loss probability at each hop.

- Scenario 2 (Localized hotspot scenario): There is a hotspot covering nodes 6 and 7 where both of these nodes experience worse conditions compared with the rest of the nodes in the network which have FER=0. This is illustrated in Fig. 4.

Table I shows the MAC retry limit, \( r \), for the different MAC-layer reliability level. For the same MAC reliability level, \( r \) increases as the physical layer frame error rate (FER) increases. This is because the MAC layer has to retransmit more in order to maintain a higher MAC reliability. Also, the for the same FER value, the value of \( r \) increases with the MAC reliability level for the same reason.

\[
 r \leftarrow \max \left\{ 3, \frac{\log \Pi - \log p}{\log p} \right\} 
\]

\( \Pi \) is the MAC layer error rate, \( p \) is the physical layer frame error rate, and \( r \) is the MAC retry limit.

\[
 R = 1 - \Pi 
\]

In this study, we fixed the value of \( \Pi = 90\% \).

<table>
<thead>
<tr>
<th>R</th>
<th>FER &lt; 0.3</th>
<th>FER = 0.5</th>
<th>FER = 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>90%</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>95%</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Fig. 2. Example of the Enhanced NACK Repair Mechanism

Fig. 3. Scenario 1: Global Hotspot

Fig. 4. Scenario 2: Localized Hotspot
TABLE II
SIMULATION PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network topology</td>
<td>Linear chain</td>
</tr>
<tr>
<td>Packet size</td>
<td>500 bytes</td>
</tr>
<tr>
<td>Number of packets(pktno)</td>
<td>300</td>
</tr>
<tr>
<td>DTSN EAR interval</td>
<td>200 msec</td>
</tr>
<tr>
<td>Routing protocol</td>
<td>Static</td>
</tr>
<tr>
<td>MAC protocol</td>
<td>802.11b</td>
</tr>
<tr>
<td>MAC retry limit (default)</td>
<td>3 (default)</td>
</tr>
<tr>
<td>PHY error model</td>
<td>Binary Symmetric Channel</td>
</tr>
<tr>
<td>Max. simulation time</td>
<td>2,000 seconds</td>
</tr>
</tbody>
</table>

V. RESULTS

A. Optimal Transmission Window

We studied the effect of the transmission window by varying the DTSN acknowledgment window (AW) size. In order to simplify our network configuration, we assume that all intermediate nodes have the same cache size (CS) and we vary the AW. We obtained both goodput and transmission cost as performance metrics. The transmission cost is the average number of link-wise packet transmissions including control and MAC layer packets and computed as follows:

\[ \text{tx\_cost} = \frac{N_{data} + N_{ack} + N_{nack} + N_{mack}}{\text{pktno}} \]  

where \( N_{data} \) is the total number of data packets transmitted, \( N_{ack} \) is the total number of transport-layer ACKs, \( N_{nack} \) is the total number of transport-layer NACKs, \( N_{mack} \) is the total number of MAC-layer ACKs, and \( \text{pktno} \) is the total number of packets that need to be delivered end-to-end.

Fig. 5 compares the goodput for CS=10 and CS=20 under scenario 1. For CS=10, the optimal goodput is achieved at \( AW_{opt}=[10,20] \) while for CS=20 optimal goodput is achieved at \( AW_{opt}=[20,30] \) as shown in Figs. 5(a) and 5(b), respectively.

In terms of energy efficiency, we see that the transmission cost is minimized in the same corresponding range of AWs as shown in Figs. 6(a) and 6(b). For scenario 2, the results follow the same pattern. For the sake of brevity, we will omit the corresponding graphs.

B. Comparison with other transport protocols

In this section, we present the overall comparative performance results in terms of goodput and energy efficiency.

We designate the different transport protocols as follows:

- DTPA – The DTPA protocol as described in [6].
- DTPA-CWL – The DTPA protocol with the transmission window set to the value of the wireless bandwidth-delay product (BDP) [12].
- DTSN+ – The DTSN protocol with the enhanced NACK repair and adaptive MAC retry limit mechanisms [10].
- TCP – The TCP protocol without the RTO exponential backoff.

For scenario 1, it can be seen from Fig. 7(a) that DTPA-CWL outperforms DTPA at FER \( \leq 0.3 \). Whereas, for FER > 0.3, DTPA-CWL provides comparable performance with DTPA. On the other hand, TCP performs closely with DTPA-CWL for FER \( \leq 0.1 \). At higher FERs, TCP begins to perform poorly compared with DTPA and DTPA-CWL. This can be explained by the fact that TCP performs AIMD-style congestion window adjustment as part of its congestion control function and as a result it increases the cwnd but ends up inducing congestion before it lowers the cwnd. DTPA and DTPA-CWL perform better at higher FERs compared with TCP because they have less overhead than TCP in terms of ACK packets and with the fixed transmission window.

Table III shows the goodput performance gain achieved by DTSN+ as compared with the other protocols. DTSN outperforms DTPA-CWL, DTPA, and TCP by as much as 720%, 723%, and 1221%, respectively, at FER=0.7. These values simply shows that the competing protocols become non-functional at high packet error rates. However, it is very significant to observe that even in such extreme FER, DTSN+
Scenario 1 – Transmission Cost, as a function of AW

For scenario 2, the goodput performance follows the same pattern as in scenario 1. Although in this case, DTPA-CWL, DTPA, and TCP+ are able to deliver the total payload of 500 packets within the maximum simulation time of 2,000 seconds for FER=0.7, albeit at very low transfer rates, i.e., 6.55 packets/sec, 7.97 packets/sec, and 2.97 packets/sec, respectively, while DTSN+ obtained a rate of 29.21 packets/sec.

In terms of goodput performance gain, DTSN+ outperforms DTPA-CWL, DTPA, and TCP+ by as much as 346%, 266%, and 883%, respectively, as shown in Table IV.

Fig. 8 shows the evolution of the TCP congestion window (cwnd) under different frame error rates. A cwnd value of 0 indicates that the end-to-end data transfer is complete. As the FER increases, the transfer completion time increases. It is evident that high error degrades the value of the cwnd up to a point where TCP only sends one packet. Moreover, at FER=0.7, DTPA-CWL, DTPA, and TCP all exceed the maximum simulation time of 2,000 seconds and are deemed unsuitable while DTSN+ continues to perform reliably.

Fig. 9 shows the packet reception graphs for scenario 1. Clearly, it shows that overall DTSN+ delivers packets faster to the destination while TCP suffers the worst performance in
higher frame error rates (i.e., $FER \geq 0.3$). Take note that in this study, we have disabled the exponential RTO backoff of TCP. If we were to enable it as a normal TCP would, we can surely expect TCP to perform a lot more poorly than presented here. For brevity purposes, we will not show the packet reception graphs for scenario 2.

Energy Efficiency

Fig. 10(a) shows the comparison of transmission cost incurred by the protocols we considered in this study. It shows that TCP closely follows the energy efficiency of DTPA-CWL up to $FER=0.30$. At higher FERs, TCP becomes more inefficient compared with DTPA-CWL. On the other hand, DTPA which uses a transmission window of 6 packets [6] is more energy inefficient compared to DTPA-CWL which uses a transmission window of 2 packets. Overall, DTSN$^+$ is the most energy efficient.

In terms of energy efficiency performance gain, DTSN$^+$ outperforms DTPA-CWL, DTPA, and TCP by as much as 39%, 65%, and 49%, respectively. For $FER=0.7$, we indicate the performance gain of DTSN$^+$ as $\infty$ to denote that DTSN$^+$ continues to perform well while the other protocols can no longer operate.

For scenario 2, the transmission cost graphs follow the same pattern as in scenario 1. In this case, TCP only begins to deviate from the energy efficiency of DTPA-CWL at $FER>0.5$. As shown in Fig. 10(b), DTSN$^+$ is the most energy efficient.

VI. CONCLUSION

The development of caching based transport protocols such as RMST, PSFQ, DTC and DTSN motivated the need to revisit
the transmission window size optimization as well as improve loss recovery semantics since both have strong impact on the effectiveness of the cache. While DTPA sets the transmission window as a function of the bandwidth-delay product (BDP), we have shown that this no longer holds with protocols that leverage intermediate caching.

This paper has presented several major contributions, namely (1) an enhanced O(1)-time complexity NACK repair mechanism; (2) experimental verification of the optimal DTSN acknowledgment window (AW); and (3) a comparative performance evaluation of DTSN with the well-known traditional transport protocol TCP and a newly proposed protocol DTPA.

Our results show the the enhanced DTSN (DTSN+) outperforms TCP and DTPA in both goodput and energy efficiency performance. This makes DTSN+ very reliable and suitable for implementation in WSNs. Our future work shall include testing these mechanisms in more complex and dynamic network scenarios as well as analyzing the performance in the presence of network congestion.

ACKNOWLEDGMENT

This work is partially funded by the University of the Philippines through the UP ERDT Program and by the Portuguese FCT SFRH/BD/75793/2011 Grant.

REFERENCES


**TABLE V**

<table>
<thead>
<tr>
<th>FER, TCP</th>
<th>DTPA-CWL</th>
<th>DTPA</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>29%</td>
<td>58%</td>
<td>29%</td>
</tr>
<tr>
<td>0.10</td>
<td>29%</td>
<td>58%</td>
<td>29%</td>
</tr>
<tr>
<td>0.30</td>
<td>25%</td>
<td>63%</td>
<td>27%</td>
</tr>
<tr>
<td>0.50</td>
<td>25%</td>
<td>57%</td>
<td>25%</td>
</tr>
<tr>
<td>0.70</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

**TABLE VI**

<table>
<thead>
<tr>
<th>FER, TCP</th>
<th>DTPA-CWL</th>
<th>DTPA</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>24%</td>
<td>28%</td>
<td>28%</td>
</tr>
<tr>
<td>0.10</td>
<td>26%</td>
<td>40%</td>
<td>28%</td>
</tr>
<tr>
<td>0.30</td>
<td>23%</td>
<td>63%</td>
<td>27%</td>
</tr>
<tr>
<td>0.50</td>
<td>25%</td>
<td>57%</td>
<td>25%</td>
</tr>
<tr>
<td>0.70</td>
<td>31%</td>
<td>48%</td>
<td>49%</td>
</tr>
</tbody>
</table>