Transparent IP Proxy for Tactical Ad hoc Networks

Helder Marques, Jérémie Leguay, Hicham Khalifé, Vania Conan, Damien Lavaux
Thales Communications & Security
4 rue des Louvresses, 92230 Gennevilliers, France
Email: firstname.name@thalesgroup.com

***Abstract*—This paper presents an adaptive and controllable framework to optimize the transport of IP packets in military MANETs. The HBH (Hop-By-Hop) protocol that we propose uses a combination of hop-by-hop reliability and congestion mitigation mechanisms. Its major advantage is to run any standard IP application unmodified on top of the network. Moreover it is able to improve flow performance in terms of latency or goodput. HBH is tunable and provides a target level of hop-by-hop reliability. It uses a transmission window and selective acknowledgements to improve bandwidth usage. We present the HBH protocol components, mechanisms and parameters. We carried out an implementation of HBH very close to a Linux implementation and validated it in the NS3 simulator. Finally, we demonstrate how HBH can improve TCP performance on a four-node chain topology by providing the right amount of additional reliability that the end-to-end TCP connection requires to overcome lossy conditions.**

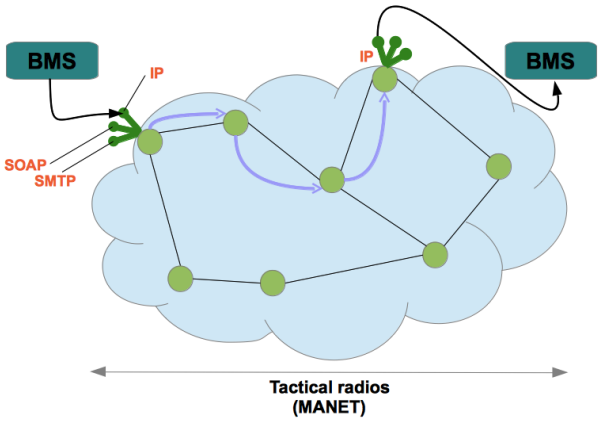


Fig. 1: Connecting applications to tactical MANETs

I. Introduction

Mobile ad hoc networks (MANET) are dynamic networks composed of mobile devices connected wirelessly. Each node can move freely and communicate with the other nodes in radio range. Multiple radio hops may be necessary before a message reaches its destination. This type of architecture has hence the advantage of being very flexible and infrastructure-less, making it robust and useful for natural disaster recovery or military purposes. The next generation of military MANETs will support new forms of operational engagement, such as Network Centric Warfare [13]. Massive transformation programs are following this path in the US [4] and in Europe [9]. Products and waveforms such as FlexNet [12], Falcon III/AN [3], or ESSOR [2], are either under development or starting to be deployed.

Battlefield Management Systems (BMS) [1] are the most common type of applications that a tactical network would run, supporting message exchanges and distributing orders. As Fig. 1 shows, these applications may either use dedicated messaging interfaces to the radio through the SMTP or SOAP protocols, or use a standard IP interface. Most of the radios developed so far have been thus offering vertically integrated messaging interface. However, this reduces the interoperability with applications and other networks. On the other hand, IP and its extensions for group communications (multicast or Xcast) offer a standard mean to address military devices. In the latter case, a BMS or any data service needs to rely on standard transport protocols (TCP or UDP) that have shown major drawbacks in MANETs in general. The frequent link failures due to mobility, packet losses due to interferences or jamming, and network capacity variations due to fluctuating radio conditions, degrade considerably TCP performance, because of its end-to-end reliability and congestion control principle, whereas UDP lacks of any reliability control loop that can compensate lossy network conditions.

To address these issues, this paper presents an adaptive and controllable framework, called HBH (Hop-By-Hop), to optimize the transport of IP packets in military MANETs. This solution aims at improving flow performance in terms of latency or goodput while being compatible with standard IP implementations to run unmodified applications on top of the network. Most of IP applications are using the TCP or UDP transport protocols. In practice, IP transport protocols fall into two extreme behaviors which are not flexible enough for tactical MANETs. On one hand, TCP offers congestion control and reliability mechanisms but has been designed for low latency and low packet loss networks. Its ACK based mechanism for full reliability makes it behave very poorly in lossy situations. On the other hand, UDP protocol does not use any feedback mechanism and is thus unable to ensure any type of control (i.e. reliability, congestion, fairness, etc...). HBH works transparently at the IP level in a hop-by-hop fashion to complement transport layer mechanisms that are end-to-end by definition. It provides a controllable level of hop-by-hop reliability and uses a transmission window and selective acknowledgements to improve bandwidth usage. The level of reliability can be controlled and chosen according to the nature of each IP flows. HBH also includes a hop-by-hop congestion mechanism to dynamically contain congestions outside of the

radio network. It can be used with UDP for tunable reliability or with TCP to hide transient link quality degradation to the source. We focus on the latter case in this paper.

The remainder of the paper is structured as follows: we first present the design choices of our HBH protocol and detail its positioning with regards to state of the art protocols in Sec. II. We then present in Sec. III its implementation in the NS3 simulator which is very close to a Linux implementation. We then study more specifically the optimization of a TCP flow over a four-node chain topology in Sec. IV. We show that HBH can provide the right amount of additional reliability that the end to end TCP flow requires to overcome lossy conditions. We present several simulation results to analyse the influence of parameters such as the window size or the desired reliability level, and to measure the impact of different RTO calculations. We finally conclude and give some perspectives in Sec.VI.

II. Transparent IP proxy

Military MANETs operate in challenging conditions, mainly due to mobility and radio constraints. Nodes have limited communication range and share the same wireless medium with their neighbors, which result in a reduction of available network resources, and adds interferences and noise issues. All these restrictions end up engendering high bit error rates and low transmission rates. These network degradations are inevitable. Our solutions aims at hiding them to transport layer protocols and at reducing miss-reactions of end-to-end transport protocols.

A. Architectural choices

Significant research has been done in the transport area, especially trying to adapt or improve TCP, the most complex transport protocol, in wireless networks. A number of lessons and ideas from this literature have been guiding the creation of our network level solution.

Standard TCP (new Reno) has been designed for very low error rates. Its flow control mechanism therefore assumes that every packet loss is due to a congestion. As a result it wrongly decreases its throughput in wireless environments. Two kinds of improvement solutions have been developed: those who attempt to tune TCP, for example through the use of Performance Enhancing Proxys (PEP) [7], [8], and those who propose a completely new protocol, which offers the same services as TCP.

Improvements of TCP can be categorized into two main ideas: *split of transport connexion* and *cross layer interactions*. The first one generally prevents the source from reducing its congestion window in case of packet losses [18]. The source is spoofed into thinking that the packet has been received (fake ACK). Similar strategies include I-TCP or M-TCP [14]. This solution may have the disadvantage of losing the end-to-end connection. In the second class of solutions, some low-layer information is forwarded to the upper layers, so that TCP knows when an error is not caused by congestion. This solution can be seen as a violation of the layering principle, but has been proven to be very effective. There are different methods to provide this cross-layer improvement: Explicit Link Failure Notification (ELFN) [11] or Ad-hoc TCP (ATCP) [16].

Completely new protocols and architectures have also been proposed for challenging networks. The Delay Tolerant Networking (DTN) [10] architecture defines a messaging service using the store, carry and forward principle. Hop-by-Hop TCP [15] and the Hop [17] protocols provide TCP-like services with an efficient decoupling of end-to-end and hop-by-hop operations. We further compare our approach to Hop in Sec. V. While being adapted to challenging conditions, these proposals require adaptations from applications.

In our case, we are able to find an happy medium between these two extremes, for an end result architecture similar to distributed PEPs: since we control and can deploy our solution in all the nodes of the network, we have the flexibility of creating a new and more fit protocol in between intermediates nodes, all the while giving a standard interface to the applications. One of the important design choices is the layer at which our solution should operate (network or transport layer). Both solutions have their advantages and drawbacks, and both are viable, the choice depending on the desired results.

The network level solution has the advantage of being able to be used indifferently by any transport protocol (TCP, UDP, ...), without being concerned about the specificities of upper protocols. The general idea is to make the network more robust to link errors and thus make it embrace the same characteristics a classic network would, hence reducing the misconceptions made by TCP. Moreover, such a solution could seamlessly be incorporated into other network mechanisms, such as IPsec. The transport layer solution has the advantage of being at a upper level, and hence has a more global view of the communication, and consequently reacts more efficiently to the issues it faces. In essence, it would allow this solution to access TCP data and even modify it if needed.

This paper presents an adaptive and controllable solution at the network level to improve the transport of IP packets.

B. Key mechanisms

Our objective is to improve flow performance in terms of bandwidth, latency and fairness, all the while not disrupting the way applications work. Technically, this is done by interfering between Layer 3 and Layer 4 to add little information in packets meant to improve the choices made by the network. As explained earlier, some transport protocols adapt poorly to the specificities of MANETs, and therefore our solution aims to increase the (perceived, from the transport protocol point of view) reliability of the MANET. In addition, in case the congestion, we prevent the unnecessary use of internal network resources by pushing the losses back to the source of flows. As a result, our solution is composed of two main hop-by-hop mechanisms:

1) Hop-by-hop reliability
2) Hop-by-hop back pressure

The general idea is hence to mimic a well known reliable protocol, TCP, but modifying its internal mechanism so that

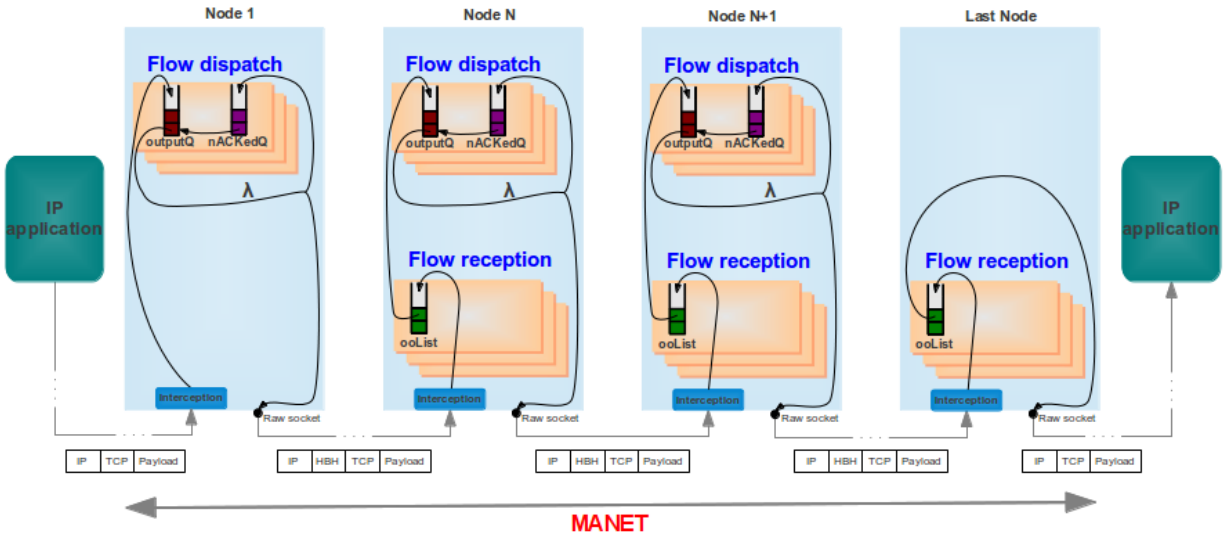


Fig. 2: HBH flow processing

there is a separation between link errors and congestion, in order to react to them more effectively.

Given that not every transport protocol or application might require the first mechanism to its full extent (*e.g.* real time applications), an option (in the form of a metric ρ) is offered to tune up the strength employed to provide this feature. However, the second mechanism is not optional, even if it implies a slight overhead which might be individually unwanted by some transport protocols, because it will reduce the overall network resource usage and will therefore indirectly upgrade the general performance of our MANET.

We named our solution *HBH*. *HBH* is made of two distinguishable parts: the first one is meant to intercept a packet when it enters the MANET, and similarly deliver it when it leaves the MANET. Further processing can be done here, in order to increase the transport protocol capabilities, through the use of PEPs for instance. The second part, which constitutes the most significant work of *HBH*, is made of a Hop-By-Hop transport by the nodes in the network, using our own protocol, which relies on IP and the routing in place. This second part will be presented in the next section.

III. The HBH protocol

As presented in Fig. 2, the *HBH* protocol intercepts all IP packets (which are unicast in this work) that arrive to the MANET or that are created within the MANET itself. It then transforms them in a *HBH* messages by adding the *HBH* protocol header, which is inserted between the layer 3 and layer 4 header. Similarly, the last hop of the MANET needs to remove the *HBH* header before passing it on. As *HBH* is not a transport protocol, a raw socket has to be used to transmit packets to the next hop. Deciding on the next hop is left to be done by the underlying routing protocol.

The data flow is processed unidirectionally, from an ingress node to a egress node, and locally from a node N to a Node $N+1$, henceforth respectively called *sender* node and *receiver node*. The are two types of messages: *HDM* (*HBH* data message) and *HAM* (*HBH* acknowledgment message). For simplicity, both of them have the same structure, described below in Fig. 3. In the rest of this section, we will describe the specification of our protocol and the format of our protocol packet.

A. Hop-by-hop reliability

The hop-by-hop reliability mechanism is ensured by robust retransmission until either the transmission is successful, or the robustness reaches the expectancy of the ρ parameter (for ease of implementation, ρ is only influenced by the maximal number of retries R). This parameter is used to finely provide an adequate level of robustness. However, there is no reordering required, so out of order received packets are sent to the next hop without delay.

Moreover, *HBH* offers the ability to acknowledge multiple packets at a time, in order to reduce the control messages required. *HBH* uses therefore a sliding window protocol. The window size W is measured in packets, and is aimed at only controlling the reliability (not the throughput, which is regulated by the back-pressure mechanism), defining the maximum number of packets not acknowledged at a time. After the transmission or retransmission of W packets, the RACK (Request ACKnowledgment) flag is set to ask for an acknowledgement. At each transmission, a RTO is calculated to trigger a later retransmission. W can be updated at each RTT using an AIMD mechanism in order to better synchronize retransmissions and the reception of acknowledgement. It has a minimal and maximal value, which for ease of implementation will be considered equal in the first version of this protocol. Finally, a selective acknowledgment feature is implemented to reduce acknowledgments overhead and cope with out-of-order reception of messages.

Offsets Byte	Byte	0								1								2								3							
Byte	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	TPE	RQT	*OptionValue*						*SeqNb*								*FlowID*															
4	**32**	*checksum*																*OrigProto*								*Reserved*							
8+	**64+**	*Data*																															

Fig. 3: HBH protocol header

B. Hop-by-hop back-pressure

The back-pressure mechanism works by sending a special message control from the receiver node towards the sender node (in the form of a *HCN* (Hop-by-hop Congestion Notification) flag inside a *HAM*), indicating that the packets are not going out of the receiver node as fast as they are coming in, causing the packets to accumulate on the receiver node, and inducing congestion. This back-pressure message is sent to the sender node whenever a threshold s of messages waiting to be sent is reached. Nevertheless, the messages coming in are not dropped until the output message queue length reaches its maximum size L ($\geq s$). An additional lower threshold could be set in place to explicitly notify the sender node that the congestion is over (not implemented in this version of *HBH*).

On the sender side, receiving a *HAM* with the *HCN* flag set, forces the emitter to reduce its sending rate, according to its internal parameter λ, which regulates the number of packets sent per unit of time.

Two distinct modes of congestion avoidance can be defined. During the initial phase, the data is sent as fast as the Layer 2 permits. As soon as the first back-pressure message is received, we enter the permanent mode, in which λ limits the throughput. λ is modified by an Additive Increase Multiplicative Decrease (AIMD) mechanism when it enters the permanent mode, and is initialized to the instantaneous throughput at that time (which can be estimated by a moving average).

C. Protocol implementation

Fig. 2 presents the overall process of packets in HBH. The flow management is divided in two parts: the one in charge of receiving the packets, and the one in charge of sending them. Each flow (as identified by its *flowID*) has its own queues.

Each flow reception module is composed of a list that maintains the sequence number of packets that have been received out of order, called *ooList*. This list is required to send *HAM*s with appropriate values. The packet themselves are however not kept in this queue, but immediately forwarded to the next hop, as there is no reordering in *HBH*.

The flow dispatch module is composed of two queues : the output queue and the non acknowledged queue. The output queue (*outputQ*) stores packets that are ready to be sent, but that can't yet be sent because of the window size limit or because they would exceed the instantaneous throughput allowed by λ, the internal parameter that controls the sending rate depending on the perceived congestion. The non acknowledged queue (*nACKedQ*) stores a copy of the packets that have been sent but non acknowledged, in case a retransmission is required.

Fig. 3 presents the *HBH* header which contains the following variables:

- **TPE**: *TyPE* of the message (*HDM* or *HAM*).
- **RQT**: Special *ReQuesT* from the packet. Represents the RACK (Request ACKnowledgment) flag for a *HDM*, and HCN flag for a *HAM*.
- **OptionValue**: SACK option for a *HAM*. A SACK option is set to 0 if not used, so there are at most 63 intervals that can be selectively acknowledged.
- **SeqNb**: The Sequence Number for a *HDM*, and the ACK Number for a *HAM*.
- **FlowID**: A unique flow identifier defined hop-by-hop. It is computed based on the 5-tuple (IPSrc, IPDest, PortSrc, PortDest, TransportProtocol).
- **Checksum**: Checksum of both data and payload.
- **OrigProto**: Identifier of the original transport protocol.
- **Data**: Data payload for a *HDM*, and the optional SACK values (if OptionValue is set) for a *HAM*.

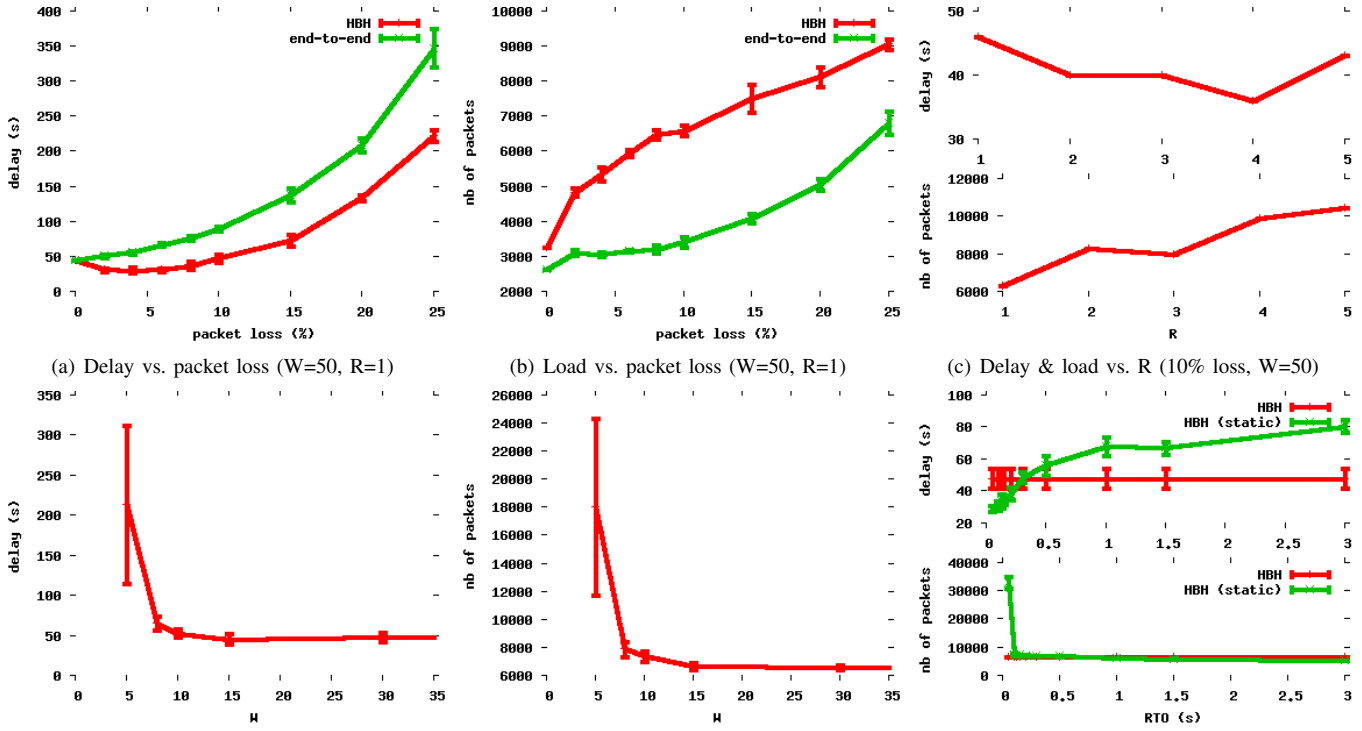
At each packet reception of an acknowledgement the RTT and hence the RTO are estimated in a similar way as TCP does. This calculation includes two constants $g = \frac{1}{3}$ and $h = \frac{1}{3}$ to define the relations between old and new RTT, and two constant $n = 10$ and $m = 2$ to define the relation between RTT and RTO. The calculation is performed as follows (M being the new RTT measure):

$$\begin{array}{rcllll} srtt & \leftarrow & (1-g) & srtt & + & g & M \\ rttvar & \leftarrow & (1-h) & rttvar & + & h & |M - srtt| \\ RTO & \leftarrow & & m * (srtt & + & n & rttvar) \end{array}$$

The computation of the $\boldsymbol{\lambda}$ parameter (in bit/s) which limits the throughput towards the next hop in congestion avoidance mode includes two constants p and q to define the relations between old and new $\boldsymbol{\lambda}$. It is set by AIMD:

$$\lambda \leftarrow \begin{cases} \lambda + p & \text{at each RTT} \\ \lambda \cdot q & \text{at each HCN received} \end{cases}$$

A *HDM* or *HAM* is made of a header and a payload, and all *HDMs* are kept by the sender node until acknowledged by the receiver node. Two internal parameters *nbTx* and *stime* are kept by the sender node to remember respectively the number of retransmissions already sent for that packet (*c.f.* the parameter ρ), and the time at which the last retransmission was sent, which helps to set up the correct timeouts.



(a) Delay vs. packet loss (W=50, R=1)

(b) Load vs. packet loss (W=50, R=1)

(c) Delay & load vs. R (10% loss, W=50)

(d) Delay vs. W (10% loss, R=1)

(e) Load vs. W (10% loss, R=1)

(f) Delay & load vs. RTO (10% loss, W=50, R=1)

Fig. 4: Simulation results for a TCP flow on a four-node chain topology.

IV. Protocol evaluation

We have implemented HBH in the NS3 [6] simulator in a very close form to a Linux implementation. A HBH module is running at each mobile node to intercept, send or release packets using a combination of Netfilter [5] hooks and RAW sockets transmissions.

In order to evaluate HBH in a more controlled environment, we consider a chain topology in which wired full duplex links are established between nodes. Thanks to this wired topology, we can finely control packet losses, transmission delays and data rates. In the rest of the paper, we consider a four-hop chain topology with links of 50ms delay and 300 kbps rate in both directions. This artificially reproduces a situation in which a distributed TDMA would have perform a uniform resource allocation. Our implementation can also run on top of a wireless network running the OLSR protocol. TCP or UDP traffic can be generated to emulate user applications. We consider in our simulations, a single TCP transfer of 100KB from the first node of the chain to the last one. Fig. 4 presents the simulation results.

Fig. 4(a) and Fig. 4(b) show the delivery delay and the network usage (in terms of number of IP packets exchanged) for HBH and the non optimized TCP connection. Results are averaged over 5 simulation runs and present 95% confidence intervals computed with the Student's t-distribution. We observe in 4(a) that the delivery delay of HBH is approximately 2 times lower than *end-to-end*. The price to pay is the network usage which in turn doubles for HBH because of the hop-by-hop acknowledgments that it uses. For high packet loss situations (greater than 10%), the network usage of HBH increases linearly while the one of TCP increases exponentially. Between 0% and 10% packet losses, HBH performs better than end-to-end at 0% packet loss. This indicates that the RTOs calculated by HBH lead to some over reactions. This may be reduced by choosing different constant values involved in the RTO calculation at a price of a higher delivery delay.

Fig. 4(d) and Fig. 4(e) show the impact of the window size W on performance. The lower W is, the closer HBH becomes to a normal ARQ (Automatic Repeat-reQuest) which repeats packets until a transmission has been successful or the maximum number of retransmissions has been reached. Such a blocking scheme does not take advantage of the full channel capacity, especially in military MANETs where packet losses can be high. The window size W (in number of packets) has then to be sufficiently high to allow parallel transmissions and retransmissions. The difficult part is to reduce acknowledgements when large values of W are used. Indeed, acknowledgements are requested at the end of each transmission window, which may lead to high values of RTOs and issues when the acknowledgment request is lost. This last packet should be repeated with a shorter RTO and the receiver should also send gratuitous acknowledgments if no packet has arrived recently.

Fig. 4(c) presents results when the threshold R is increased. R defines the maximum number of times a packet can be retransmitted. This value is maintained hop-by-hop but could

be transported and used at end-to-end connection scale using the *reserved* field in the HBH header. This result shows that the delay is minimum for R=4. Indeed, increasing the reliability hides packet losses to TCP. However, trying to achieve full or too high reliability on behalf of TCP leads to poor performance. The reason for this is that packet delay may exceed TCP's RTO in some cases. This leads to duplicates of the same TCP segment that HBH has to handle at the same time but at different places.

Fig. 4(f) compares the performance of HBH with a slightly modified version of itself where a static value for RTOs is used. In this simulation, the RTT on 1 hop is around 110 ms. Results show that for static RTOs below this value, lower delays could be obtained as it accelerates artificially packet transmissions at very high overhead and fairness costs. On the other hand, static RTOs much higher than this value reduce the overhead a lot while leading to poor delays. This result highlights the benefits of our dynamic RTO estimation but also emphasises the importance of choosing good constant parameters in its calculation.

V. Related work

We believe Hop [17] to be the solution closest to ours. However there are many conception design differences, which we highlight in this section.

Perhaps the easiest difference to spot is that these two solutions do not work at the same level. Hop is a layer 4 protocol specifically designed for MANETs. HBH, however, is a layer 3.5 solution, so that standard transport protocols may work over HBH. In this way, applications do not need to be modified and may use their usual sockets. One of the consequences of this design choice is that HBH may not propose 100% reliability. It offers the opportunity to tune the desired reliability depending on the level required. It can be tuned for example so that a TCP traffic may not suffer too much from high losses or so that a real time application may not pass unusefully a delivery dead line.

Unlike Hop, HBH is a connectionless protocol, in the sense that there is no SYN/ACK mechanism. The virtual retransmission mechanism in Hop is hence not used in HBH. There is also no end to end ACK mechanism (unless implemented by the transport protocol). In case of routing changes, new link sessions will be created on the fly. As opposed to Hop, HBH does not provide a cache that remains in the node even after a hop-by-hop acknowledgment has been received.

Finally, Hop works in a block scheme, whereas HBH provides a flow mechanism in a packet switching network. Therefore, HBH uses a sliding window mechanism, and a selective acknowledgment method, that is not used by Hop. The back pressure mechanism is also different: in Hop, the congestion is inferred from the non reception of an ACK. In HBH, there is an explicit message to inform the previous node of the congestion.

VI. Conclusions and Future directions

This paper has presented an adaptive and controllable framework to optimize the transport of IP packets in military MANETs. This solution aims at improving flow performance in terms of latency or goodput while remaining fully compatible with standard IP implementations to run unmodified applications on top of the network. We have proposed the HBH (Hop-By-Hop) protocol which uses a combination of hop-by-hop reliability and congestion mitigation mechanisms. We have presented the extensive implementation work that we realized in the NS3 simulator to evaluate HBH performances. Our simulation results show that HBH can provide the right amount of additional reliability that end to end TCP flows require to overcome lossy link conditions. We presented several simulation results to analyse the influence of parameters such as the window size or the desired reliability level, and to measure the impact of different RTO calculations.

Future work includes large scale testing to evaluate performance of HBH flows in terms of fairness, delay and overhead when mixing various end-to-end TCP and UDP connections. Different methods for RTO calculations in lossy environments should also be evaluated to achieve the best latency while keeping the overhead low.

References

[1] Battlefield Management System. http://www.thalesgroup.com/Portfolio/Defence/LandJoint_Products_CommandControl_tactical_T-BMS/.

[2] European Secure SOftware defined Radio. www.occar.int/28.

[3] Harris AN/PRC-117G wideband tactical radio. http://rf.harris.com/capabilities/tactical-radios-networking/an-prc-117g/.

[4] Joint Tactical Radio System Homepage. http://jpeojtrs.mil.

[5] Netfilter/iptables project homepage. www.netfilter.org.

[6] Network simulator 3 - ns3. http://www.nsnam.org.

[7] Performance Enhancing Proxies (PEP) Intended to Mitigate Link-Related Degradations. RFC 3135 (Proposed Standard).

[8] TCP Performance Implications of Network Path Asymmetry. RFC 3449 (Proposed Standard).

[9] Le GTIA Renouvelé, Dossier Scorpion. *Proc. Terre Magazine N 215 - In French*, June 2010.

[10] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. ACM SIGCOMM*, 2003.

[11] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proc. ACM Mobicom*, 1999.

[12] R. Iovine and J. Bouis. The Flexnet-Waveform in the international SDR arena. In *Proc. IEEE MILCOMM*, 2009.

[13] B. K. Haberman J. L. Burbank, P. F. Chimento and W. T. Kasch. Key Challenges of Military Tactical Networking and the Elusive Promise of MANET Technology. *Proc. IEEE Communications Magazine*, 2006.

[14] S. Singh K. Brown. M-TCP: TCP for Mobile Cellular Networks. *ACM Computer Communications Review*, 1997.

[15] Yao-Nan Lien and Yi-Fan Yu. Block-switched Networks: A New Paradigm for Wireless Transport. In *Proc. IEEE APSCC*, 2008.

[16] J. Liu and S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. *Proc. IEEE Journal on Selected Areas in Communications*, 1999.

[17] Deepak Ganesan Arun Venkataramani Ming Li, Devesh Agrawal. Block-switched Networks: A New Paradigm for Wireless Transport. In *Proc. ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[18] S.V. Krishnamurthy S. Kopparty and M. Faloutsos. Split TCP for mobile ad hoc networks. In *Proc. IEEE GlobeCom*, 2002.