Non Disruptive Data Services Towards Real-time Traffic in Wireless Ad Hoc Networks

Jérémie Leguay[1], Hicham Khalife[2], Georgios Sotiropoulos[1], Vania Conan[1], Naceur Malouch[2]
[1] Thales Communications [2] UPMC Univ Paris 06

***Abstract*—Mobile wireless Ad hoc NETworks (MANETs) naturally support a traffic mix of elastic and real-time flows but the shared nature and lossy properties of the radio medium make their coexistence challenging. We argue in this paper for a new kind of elastic data transport service which would preserve the quality of real-time priority flows while guaranteeing an acceptaple (tunable) end-to-end delivery time of elastic data. We propose to use mechanisms from Delay Tolerant Networking (DTN) to support hop-by-hop data transfer from source to destination and an adaptation of TCP which monitors its agressiveness towards local VoIP traffic on each hop. The scheme is evaluated in simulation on a simple 4-node scenario and in a more realistic case where doubling data transfer time allows for the support of seven VoIP flows of medium quality against none for standard TCP.**

I. Introduction

In *Mobile wireless Ad hoc NETworks* (MANETs) [1] wireless mobile nodes are able to communicate with one another either directly whenever they are within transmission range or using multi-hop routing.

MANETs provide natural support of multi-commodity traffic with coexistence of elastic data services and real-time constrained flows. The latter, such as VoIP applications, use transport protocols based on UDP and are sensitive to packet losses and jitter. Elastic data services are typically provided by TCP. Simple examples (see e.g.; Section II) show how much TCP can be disruptive toward VoIP in a wireless context.

One traditional approach to addressing this issue is by providing QoS mechanisms in the network. The MAC layer could use, for instance, slot allocation to priorize flows in a distributed TDMA-like scheme [2], or QIBSS (QoS independent basic service set) of the IEEE 802.11e standard in the CSMA-CA case.

Another approach is to consider TCP, and its well understood shortcomings in wireless ad hoc networks, as the root of the problem. TCP is not capable of differentiating losses from congestion, wireless errors, link failures or contention which results in poor end-to-end performances. TCP is also too aggressive since it tries, as part of its fundamental design principles, to reach the bottleneck throughput on the links used by its connection notwithstanding the interferences this may be generating [3]. Several adaptations of TCP have been already advocated such as [4], [5], [6], nevertheless, most of these proposals focus on enhancing the TCP throughput as a primary goal. To the best of our knowledge, none of them target at providing quality of service to delay sensitive flows in a wireless ad hoc network.

We argue for the study of a new kind of service for elastic data transport whose main property is to be conservative with regards to real-time services. In this paper we propose and evaluate such a new elastic data transport protocol which focuses on preserving the quality of VoIP traffic. The main contributions of the work are:

1) *Delay Tolerant framework.* To carry elastic data from source to destination, we rely on hop-by-hop mechanisms from Delay Tolerant Networking (DTN).
2) *Link-level protocol* For each hop, we employ an adapted TCP, reusing AIMD congestion control and introducing a new parameter τ which monitors the aggressiveness of TCP towards local VoIP traffic.

The remainder of the paper is organized as follows. Section II motivates and introduces the framework that we propose and section III presents the TCP adaptation. In Section IV, we show simulation results. Finally, we present our concluding remarks and identify directions for future work in Section V.

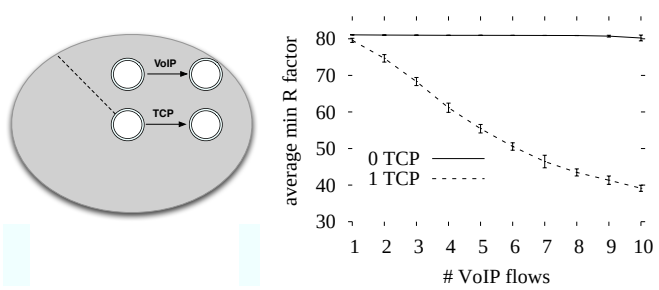
II. Non disruptive data services

This section provides an overview of the new class of non disruptive data service we advocate in MANETs.

A. Motivation example

Let's consider the simple practical scenario shown in Figure 1(a) where several VoIP flows compete with one TCP data connection over a single hop. All considered nodes are using the IEEE 802.11 protocol at the MAC layer and within the range of each others thus contending for the same resources. The perceived quality of VoIP transmissions can be measured by computing the so called $R-factor$ as standardised by the ITU [7]. An R-factor above 70 corresponds to a voice call of medium quality in a scale of 100 values (100 provides a call of perfect quality but realistic parameters, such as equipment noise, reduce its maximum feasible value to 81). As shown in Figure 1(b), in this simple scenario, no more than two VoIP calls can be accepted in the presence of the TCP data exchange. Clearly, such performance degradation is caused directly by the TCP exchange since in the absence of data transfer an R-factor of 81 is attained for at least 10 VoIP connections.

Our objective can be stated as designing a data transfer protocol which preserves the quality of priority traffic while guaranteeing an acceptable end-to-end delivery time.



(a) Scenario where all 4 nodes are within range

(b) VoIP quality with and without 1 TCP connection

Figure 1. Impact of TCP on VoIP traffic on a wireless link

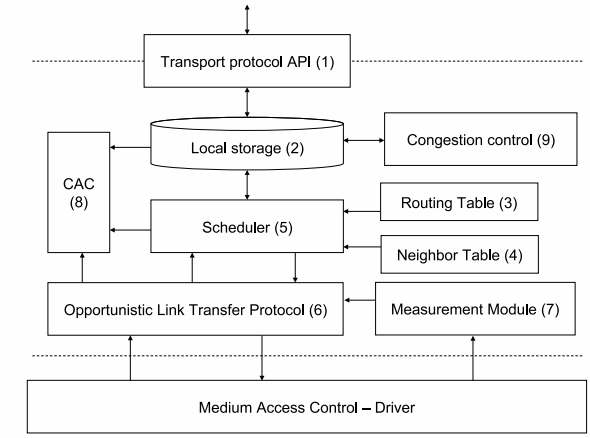


Figure 2. Transport protocol architecture

B. General idea

Our proposition is to make elastic data move smoothly towards their destinations. Data is transferred very cautiously, by constantly monitoring the impact of the data transmissions on real-time traffic. To do so, our solution acts at both a macroscopic and a microscopic level.

At the macroscopic level, the idea is to break the usual end-to-end communication paradigm. We propose to use store and forward transport, meaning that intermediary nodes along a path wait to receive the whole message before forwarding it to the next relay. Messages could be of any size (e.g., e-mail, picture). This way of transporting data exhibits certain characteristics that make it appealing for our field of applications. In contrast to end-to-end transfer, the *DTN-fashion hop-by-hop transfer* confines the interference/disruption caused to other flows spatially only in the wireless neighborhood of the current transfer hop. Moreover, intra-flow interference is minimized whereas hidden node problems within the flow caused by multi-hop transmission are not present. We expect that all these intrinsic characteristics would provide efficient controllability of the disruption caused to VoIP flows by transmitting data and contribute to the preservation of their quality[1].

Having ensured that data transmission (at the macroscopic level) only causes disruptions in the vicinity of the current transfer hop, at a microscopic level we propose to use an opportunistic transport protocol, that aims at further controlling the impact of data transfers on the VoIP flows (see Section III).

C. Transport protocol architecture

Fig. 2 presents the different entities that would run on each node to implement the transport protocol. This architecture is similar to the regular DTN stack but includes modules specific to our needs [9]. The system is composed of the following modules (the numbers match those of Fig. 2):

1) **Transport protocol API:** Applications willing to transfer data using the hop-by-hop protocol use this API to send/receive messages (called bundles).
2) **Local storage:** Nodes use this storage facility (e.g., database, files) to keep persistently messages in transit.
3) **Routing table:** This table contains the routes used to determine the next relay on the way to a given destination. This table is copied from a regular unicast IP routing protocol (e.g., OLSR) or from a DTN routing protocol (e.g., ProPhet).
4) **Neighbor table:** This table contains the list of nodes currently under transmission range. The information can be collected from the routing protocol neighbor discovery functions.
5) **Scheduler:** The scheduler decides when a locally stored message should be transferred to the next hop. It can implement various priority schemes.
6) **Opportunistic Link Transfer Protocol:** This module implements the opportunistic transfer protocol defined in Sec. III to transfer bundles on each hop along paths.
7) **Measurement module:** This module collects information regarding real-time or priority flows in the wireless neighborhood thanks to the broadcast nature for the wireless channel. Predictive quality indicators are constantly computed to serve as input by the sender module.
8) **Connection Admission Control (CAC):** Optional module that aims at accepting or rejecting requests for incoming bundle transfers.
9) **Congestion control:** Optional module that implements the end-to-end or global control loop required to regulate congestion in the overall network.

III. Opportunistic link transfer protocol

On each hop the opportunistic link transfer protocol must limit disruptions towards local VoIP flows. Delivery must also be reliable. Whereas it would be possible to propose a

[1]Other applications could benefit from the intrinsic controllability characteristics of the DTN-fashion hop-by-hop transfer. For instance, interference-aware routing which aims at routing around interference ”hotspots” in the network could benefit from the rather spatially confined interference that this way of transport induces, leading to throughput gain according to [8].

brand new one-hop transfer protocol we favored simplicity and decided to adapt TCP to suit these requirements.

A. TCP adaptation

TCP uses the additive increase/multiplicative-decrease (AIMD) algorithm for congestion avoidance. AIMD represents a linear growth of the congestion window, combined to an exponential reduction when a congestion takes place. As long as non-duplicate Acknowledgements (ACKs) are received, the congestion window is additively increased by α every Round Trip Time (RTT). When a packet is lost (or arrive out-of order), duplicate ACKs will be received. When three duplicate ACKs are received, TCP Reno, the one that we used, decreases the congestion window by multiplicative factor β, performs a "fast retransmit", and enters in a phase called Fast Recovery. If an ACK times out, slow start is used. Note here that in the current implementations of TCP the default values of α and β are respectively 1 and 0.5.

Our TCP Reno adaptation consists in dropping incoming packets at the receiver whenever the measurement module signals that the quality of the neighboring VoIP flows is altered. In fact, the AIMD procedure in this case is guided by the neighboring VoIP connections quality and not encountered congestion as in standard TCP. More practically, each node is running a VoIP measurement module which monitors all the VoIP flows that a node can overhear. Measurement modules maintain a Boolean state called VOIP_LOSS being equal to *false* if the quality of VoIP flows is good and *true* otherwise. At the TCP receiver, the following algorithm is then applied each time a packet P arrives:

1: **for** every arriving packet P **do**
2: **if** VOIP_LOSS = $true$ & last reaction older than τ **then**
3: drop P
4: **end if**
5: **end for**

Reaction time τ is the main parameter of our TCP adaptation. The higher τ, the less often does TCP reduce its congestion window in case of persistent VoIP impairments. In other words, this parameter controls TCP aggressiveness towards VoIP flows and allows to trade off elastic data delivery time for VoIP flow quality.

B. Measurement module

A VoIP measurement module is implemented on every node. This module listens to all incoming VoIP packets on the wireless interface using for instance a virtual interface in promiscuous mode or a hook in the wireless driver. VoIP packets can be identified using flow tagging techniques or by inspecting RTP headers. The measurement module then maintains a per flow history H of packets that have recently arrived.

To measure the quality of a voice call, we referred to the E-model [7] defined by the ITU, an end-to-end performance estimation tool that provides a prediction of the expected voice quality as perceived by a typical telephone user. A value of the R-factor above 70 corresponds to a voice call of medium quality, while a score of 100 provides a voice call of the best quality. A simplification proposed by Cole et al. [10] for the G.729 codec makes this model suitable for computer networking as it takes into account mouth-to-ear delay and loss rate. The R-factor is then given by the the following formula:

$$\begin{aligned} R &= 94.2 - 0.024 * d - 0.11 * (d - 177.3) * H(d - 177.3) \\ &\quad -11 - 40 * \log(1 + 10 * e) \end{aligned}$$

Where:

- $d = d_{codec} + d_{jitter\_buffer} + d_{network}$ is the total mouth-to-ear delay comprising of the codec delay due to processing and packetization (typically 25ms), the delay of the de-jitter buffer (typically 60ms) and the transfer delay of the network. The length of the de-jitter buffer is the maximum delay variation between consecutive packets. Any packet that fails to meet this delay deadline is discarded and accounted for jitter buffer losses.
- $e = e_{network} + (1 - e_{network})e_{jitter\_buffer}$ is the total loss comprising of network induced losses and losses due to the de-jitter buffer.
- $H(x) = (x > 0)?1 : 0$ is the Heaviside step function.

The calculation of the R-factor finally reduces to measuring only the packet delay and losses. The measurement module thus monitors two vital parameters for VoIP flows. First, it measures the average *jitter* (i.e., delay variation) of recently received packets. Because nodes see VoIP packets passing without being involved in VoIP sessions, real packet delay is impossible to obtain by listening to the wireless medium. However, as VoIP uses a constant packet rate, we are able to deduce from time of arrival of packets the current jitter, as if they were received locally. By trying to keep the jitter low we intend to reduce end-to-end delay as in wireless networks jitter is significantly impaired by congestion or contentions on the wireless channel. Second, the measurement module measures the *packet loss* which consists in the number of packets recently lost. This is achieved by observing sequence numbers in RTP headers.

To compute the local VOIP_LOSS state used by our TCP adaptation, the measurement module then applies the following algorithm each time a VoIP packet P is sensed:

1: F_p is the flow identifier of P
2: P is stored in $H(F_p)$
3: detect losses on packets in $H(F_p)$ using RTP seq. #
4: **for** all F_p **do**
5: **if** loss detected **then**
6: set the local VOIP_LOSS state to *true*
7: clear $H(F_p)$
8: **else**
9: compute jitter on packets in $H(F_p)$
10: **if** jitter $> \sigma$ **then**
11: set VOIP_LOSS to $true$
12: clear $H(F_p)$
13: **end if**

14: **end if**
15: **end for**

In other words, VoIP losses and VoIP jitter variations above σ produce the same effect in our link transfer protocol as losses in standard TCP. The local VOIP_LOSS state is maintained for at least θ seconds. After this duration, if no issues are observed on neighboring VoIP flows, it is set back to *false*. From their local VOIP_LOSS state and those of their 1-hop neighbors, measurement modules build an extended neighborhood VOIP_LOSS state using periodic exchanges of control information.

IV. Simulation results

To evaluate our solution, we present NS-2 simulation results. Our objective is twofold: first, show that data transfers conducted with our mechanism preserves the quality for VoIP communications (i.e., we seek an R factor $\geq$ 70) and second, ensure that data transfers are accomplished within an acceptable delivery time. We first thoroughly study how our transport protocol performs using the simple 4-node topology presented in Fig. 1(a). We then move on to apply the proposed mechanism on a realistic multihop topology.

A. Simulation parameters

We considered nodes equipped with a IEEE 802.11 wireless interface running in ad hoc mode at 2 Mbits with RTS/CTS enabled. Hop by hop transfers are carried out by a DTN daemon and the measurement module was implemented using a TAP interface. In each scenario, TCP file transfers of 5MB start 50s after VoIP flows thus making sure that DSDV [11], the routing protocol that we used, has converged. In all simulations σ=30ms and θ=20ms. We performed 20 simulation runs for each scenario and the results reported in the paper are mean results with confidence intervals at the 95% confidence level, obtained using the Student t distribution. Fig. 3 presents the simulation results.

The G.729 [12] VoIP codec uses a bitrate of 8 kbit/s. It sends 50 packets per second, each containing 20 bytes of data. We used VAD (Voice Activity Detection), meaning that no packets are transmitted during silences. To generate realistic VoIP traffic between two nodes A and B, we used the P.59 recommendation from the ITU [13] which defines a Markov chain with four states: only A speaks, only B speaks, A and B speak, both A and B are silent. We used the proposed state transition probabilities. VoIP flows last for the whole duration of the simulation experiment and for each flow we calculated the average R-factor over the whole duration. In the graphs we chose to plot the minimum average R-factor between all flows.

B. Impact of reaction time τ

The value of reaction time τ dictates the aggressiveness of the transfer protocol towards the neighboring VoIP flows. When set to a low value, and regardless of previous reactions, τ forces the transfer protocol to throw packets away, thus to reduce its sending rate based on TCP's AIMD. Whereas when τ is large the transfer protocol reacts less frequently to encountered VoIP quality variations (VoIP losses or jitter violations) hence avoiding to severely penalize the data. Fig. 3(a) and Fig. 3(d) show the reaction time effect in the 4-node scenario for simulations of 400s. Unsurprisingly, setting the value of τ to 0 achieves the best performance in terms of number of accepted VoIP flows. Note that in the formula that we used for the R-factor, a value greater than 81 is impossible due to the equipment impairment factors considered. We considered that having VoIP flows with a quality under 70 would impact the system usability.

Nevertheless, it may be of interest to set τ to 20 ms for instance. In this case up to 10 VoIP flows can be served with medium quality. Moreover the delivery time of the data transfer remains acceptable (300 s versus 50 s for 1 VoIP flow). If shorter data transfer times are required, one can set τ to 50 ms: delivery time is reduced to 100 ms and 7 VoIP flows can still be supported.

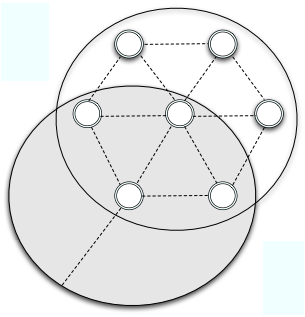


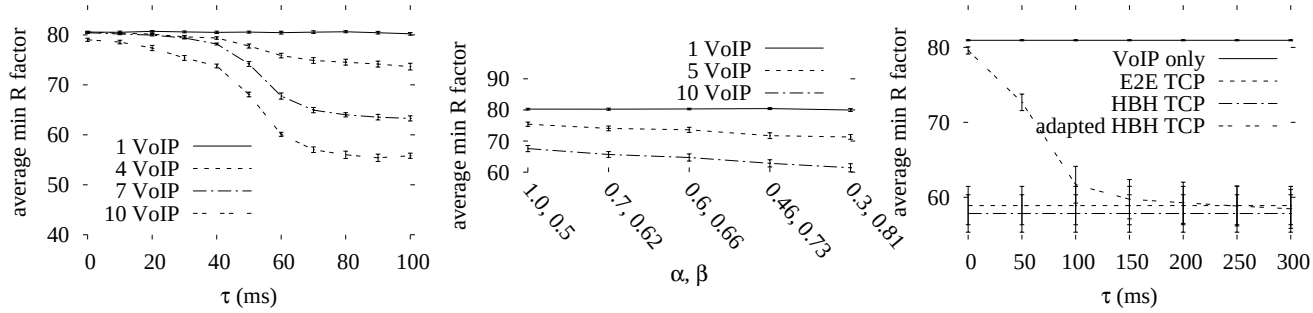
Figure 4. Dense multihop scenario

C. Impact of TCP parameters α and β

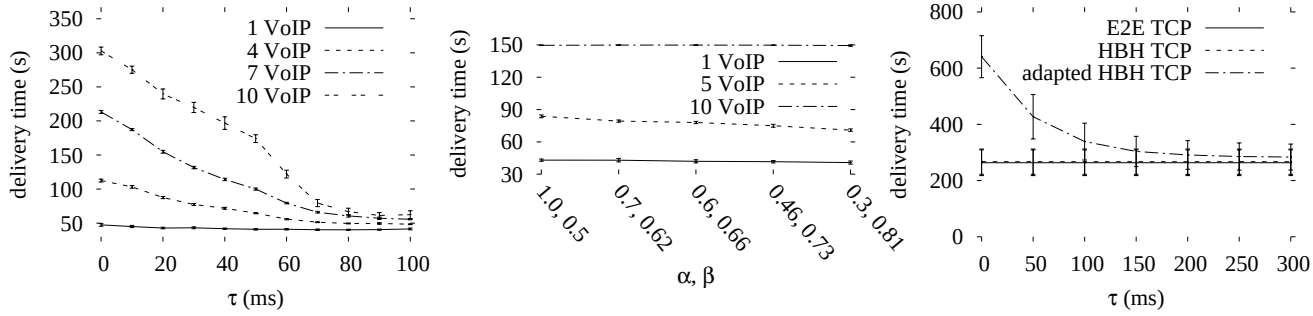
As mentioned in Sec. III-A, α and β correspond respectively to the additive and the multiplicative factors used by TCP's AIMD. Fig. 3(b) and Fig. 3(e) shows the influence of those factors for different pairs of them (i.e., [1.0, 0.5], [0.7, 0.62], [0.6, 0.66], [0.46, 0.73], [0.3, 0.81]), chosen in order to preserve TCP stability (see [14] for computation). We used a τ of 50ms and a simulation time of 200s. Surprisingly, we can observe that reducing α, or increasing β, increases the aggressiveness of TCP towards VoIP flows. As a consequence, one could further refine the one hop transfer protocol by tuning these parameters dynamically to speed up bit data transfers while keeping VoIP quality above a certain threshold.

D. Performance on a multihop topology

Fig. 3(c) and Fig. 3(f) show simulation results on the multihop topology presented in Fig. 4. Between random pair of nodes, we generate 6 VoIP flows and 3 data transfers conducted using: normal end-to-end (E2E) TCP, hop-by-hop (HBH) TCP, and adapted hop-by-hop TCP (our proposition). The simulation time is 1000s. The results show first that when no data transfers are performed, VoIP quality is good (i.e., around 80). While, when using E2E TCP or HBH TCP,



(a) VoIP flow quality (4-node scenario) (b) VoIP flow quality (4-node scenario) (c) VoIP flow quality (7-node scenario)



(d) TCP flow durations (4-node scenario) (e) TCP flow durations (4-node scenario) (f) TCP flow durations (7-node scenario)

Figure 3. Simulation results for the 4-node and 7-node scenarios

the quality is severely impacted (i.e., around 60). This shows that splitting TCP to single hop connections without further adaptation does not solve our problem. The adapted HBH TCP mechanism that we propose manages to preserve VoIP quality when τ is 0. As in previous results, VoIP quality degrades when τ increases, until it reaches the performance of E2E TCP.

V. Conclusion and future work

In this paper, we have introduced a new class of service which aims at protecting real-time priority traffic in MANETs. We proposed a framework that combines hop-by-hop data transfer from DTN and opportunistic one-hop data transfer. We envision that this approach could open up new perspectives to motivate the study of a more efficient and comprehensive elastic data transport protocol for MANET environment where TCP plots poor performance.

In this respect, a number of issues of interest remain to be further investigated. The benefits that other QoS enablers, such as MAC packet prioritisation or QoS routing, may provide could be evaluated and their mutual interplay analysed. Complementary mechanisms can be added, such as end-to-end congestion control, bundle segmentation size selection, or mixes of one-hop and multiple-hop transfers. Extensions to other real-time or priority traffic (e.g.; video) is also left for future study.

References

[1] S. Corson, "Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations," RFC 2501, 1999.

[2] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," in *Proc. ACM MobiHoc*, 2005.

[3] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Performance," *Mobile Computing, IEEE Transactions on Mobile Computing*, vol. 4, no. 2, 2005.

[4] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad Hoc Networks," in *Proc. ACM MobiHoc*, 2003.

[5] S. Kopparty, S. Krishnamurthy, and M. F. S. Tripathi, "Split TCP for Mobile Ad Hoc Networks," in *Proc. IEEE GLOBECOM*, 2002.

[6] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *Wireless Networks*, vol. 8, no. 2, 2002.

[7] "ITU-T Recommendation G.107, the E-Model, a computational model for use in transmission planning."

[8] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-Hop Wireless Network Performance," *Wireless Networks*, vol. 11, no. 4, 2005.

[9] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay tolerant network architecture, IRTF draft, draft-irtf-dtnrg-arch-02.txt," July 2004.

[10] R. G. Cole and J. H. Rosenbluth, "Voice over ip performance monitoring," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2, 2001.

[11] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, 1994.

[12] "Coding of speech at 8kbits/s using conjugate structure algebraic code excited linear prediction, ITU-T recommendation G.729 March 1996."

[13] "Artificial conversational speech, ITU-T recommendation P.59. 1993."

[14] J. Padhye, V. Fioriu, D. Towsley, and J. Kurose, "Modeling tcp throughput: a simple model and its empirical validation," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, 1998.