Prototyping of Time-Division Unbalanced Carrier Sense Multiple Access and First Experiments

Andrea Vesco, Gian Marco Toso and Riccardo Scopigno
Istituto Superiore Mario Boella (ISMB) – Torino, Italy
Email: {vesco,scopigno}@ismb.it

***Abstract*—This paper presents the work of prototyping a weakly synchronous and distributed coordination function called Time-Division Unbalanced Carrier Sense Multiple Access (TDuCSMA). The prototype, designed and developed by modification of the wireless Atheros drivers, is meant to demonstrate the feasibility of TDuCSMA and its capabilities in bandwidth management and QoS provisioning. Furthermore, the bandwidth reservation model, derived and verified by simulations in previous works, is validated by real communication experiments.**

I. INTRODUCTION

Wi-Fi is a great technology to extend access to the backbone networks of Internet Service Providers (ISP). Today, access is provided by small-scale Wi-Fi single-hop access networks and resource over-provisioning is widely used to satisfy user requirements. However, there is a market opportunity for deploying large-scale Wi-Fi multi-hop access networks in environment such as small cities, campus, sport centers and malls. In these network scenarios resource over-provisioning would not be cost-effective hence a solution for bandwidth and traffic management is required to let ISP implement the quality-of-service (QoS) models from the access and not only in the backbone network. Thus, the challenge is to design scalable and flexible MAC solutions to satisfy these requirements. The channel access has been arbitrated by either CSMA/CA or TDMA for years. In CSMA/CA based solutions the decision making process is distributed among all nodes. Each node determines individually when it is time to access the channel. Thus CSMA/CA is a distributed solution relying on the principle of random access. However, even if distributed, easy to implement and scalable, CSMA/CA suffers from a performance degradation due to collisions and sub-optimal decisions. In fact, it poorly performs when strict QoS is required. Moreover CSMA/CA performances get even worse in a multi-hop networks because (*i*) node density increases access delay and reduce the overall throughput and (*ii*) queuing and access delay at each hop additively contributes to the end-to-end delay. Alternative solutions attempt to guarantee QoS with TDMA. In TDMA based networks, nodes access the network at well defined instants, thus avoiding contentions and collisions. This approach is successfully adopted in [1], [2] where the limitation due to a centralized coordination function is also overcome and reservation is managed in a distributed fashion. However such solutions still suffer from the intrinsic stiffness of a reservation mechanism which hardly copes with rapidly changing traffic profiles; additionally they require extensions to the standardized MAC layer that are very likely to increase time-to-market and make a wide deployment difficult.

In the Author's view an alternative is represented by a weakly synchronous and distributed coordination function, called Time-Division Unbalanced Carrier Sense Multiple Access (TDuCSMA). It relies on synchronization among nodes and the time-driven switching of contention parameters inside nodes to provide a viable solution for bandwidth management over multi-hop access networks. In principle TDuCSMA provides the capabilities of reserving bandwidth to traffic aggregates throughout the network, while exploiting all the available bandwidth. The TDuCSMA operating principles have been extensively addressed in single [3] and multi-hop [4] scenarios by simulations and analytical models. Moreover the work in [5] has proved the fully compliance to the IEEE 802.11 standard and has designed an architecture enabling the coexistence of TDuCSMA and CSMA/CA entities on the same node. The TDuCSMA is flexible enough to provide to the upper layers the knobs for driving its operation, hence to implements dynamic and distributed bandwidth and traffic managements by signaling architecture [6]. As a result, ISP can implement their own management policies while providing QoS and best-effort services from end-to-end.

The promising results have motivated the prototyping of TDuCSMA. The ISP are very cautious with new technologies and need to "*see and experience*" before considering introducing them in their networks. The same is for the manufacturers. Although the prototype cannot be anything suitable for field trial by ISP, it is indeed instrumental in showing the feasibility of the proposed solution. Specifically, it proves not only that what has been proposed can actually be implemented, but also that it actually fulfills the claims on low complexity and bandwidth management together with QoS provisioning.

This paper presents the TDuCSMA prototype in details and its first tweets on the real-world.

II. TIME-DIVISION UNBALANCED CSMA

A. Operating Principles

In TDuCSMA networks all nodes are synchronized with the common time reference (CTR) structure depicted in Fig. 1. The CTR is a periodical time structure where the time-frame TF is the time unit and k TFs are grouped in a time-cycle. The time-cycle length TC – measured in TFs – provides the periodicity of the CTR structure. Both the time-frame duration T_f and TC are tunable system parameters. The synchronization can be distributed using the coordinated universal time

(UTC) to derive T_f from a global navigation satellite system (GNSS) or by a distributed leaderless solution [7], where nodes collaboratively reach a consensus on a common clock.

Even though the typical TDMA time structure is deployed, the decision making process about channel access is distributed among all nodes following CSMA/CA rules. Each node maintains two sets of EDCA parameters ($AIFS$, CW_{min} and CW_{max}) and all nodes have the same sets. These sets are referred to as *high-priority* set $EDCA^H$ and *low-priority* set $EDCA^l$. The EDCA parameters are *unbalanced* in the two sets. More formally $AIFS^H < AIFS^l$ and $CW^H_{min} \leq CW^H_{max} < CW^l_{min} \leq CW^l_{max}$ such that node i, contending for channel access in accordance with $EDCA^H$, has almost strict priority on node j using $EDCA^l$ settings.

The underlying idea is to synchronize the contextual switching of EDCA parameters at each node such that (*i*) only one node contends for channel access in accordance with $EDCA^H$ at a time and (*ii*) all nodes maintain $EDCA^H$ for a predefined periodical time interval, referred to as T_H.

Fig. 1 also shows the time-driven switching of EDCA parameters inside three nodes sharing the same collision domain. As depicted, only one node contends for channel access in accordance with $EDCA^H$ during one TF, whereas the time periods in which nodes operate in accordance with $EDCA^j$ $\forall j = H, l$ can change over the nodes. As a result of TDuCSMA operating principles, a node i is very likely to gain access to the channel and maintain it for the full period T^i_H. It worth noting that this happens due to CSMA/CA operations and the values of the access parameters in $EDCA^H$ and not because of a predefined channel access as in TDMA based solutions[1].

In principle the EDCA parameter sets are switched over time in a per-node basis, so that each node handles QoS-demanding traffic as a single aggregate. Thus, bandwidth management is performed in a per-node basis by assigning different T_H to nodes sharing the collision domains. However a sub-set of TFs can be left un-allocated to let node send background traffic in accordance with the either best-effort or differentiated service discipline as addressed in [5].

Moreover, since TDuCSMA preserves the CSMA/CA nature, if a node i does not have enough traffic to be sent before the end of its T^i_H, any other node can gain access to the channel, thanks to CSMA/CA and transmit. Hence, bandwidth re-use is easily and intrinsically implemented and bandwidth wasting, as side effect of reservation, is avoided.

B. Bandwidth Reservation Model

The work in [3] verified two important consequences of the TDuCSMA operating principles:

1) only node i gains access to the channel during T^i_H, thus the congestion windows in $EDCA^H$ can be minimized to reduce back-off time between two consecutive transmissions hence to increase bandwidth utilization without affecting collision probability;
2) if node i tends to use its T^i_H with poor efficiency due to short packets, this does not affect the transmissions of the other nodes in their respective T_H periods.

[1]The transmission opportunity *TXOP* mechanism is not exploited in TDuCSMA because if a node were delayed in its channel access, *TXOP* would enforce this delay and propagate it with a disruptive effect on the underlying TDuCSMA operating principles.

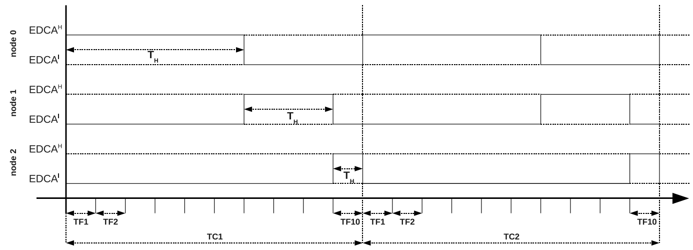

Fig. 1. Time-driven EDCA parameters switching inside three nodes; $T^0_H = 6$, $T^1_H = 3$, $T^2_H = 1$ over a time cycle with $k = 10$ TFs.

Therefore, assuming $CW^H_{min} = CW^H_{max} = 1$ and neglecting the propagation delays, the theoretic bandwidth G_{id}, available for reservation, can be expressed as the efficiency in channel utilization considering only the protocol overheads as follows:

$$G_{id} = \frac{Rt_p}{t_p + AIFS^H + 2t_{plcp} + t_h + SIFS + t_{ack}}, \qquad (1)$$

where R is the line-rate, t_p and t_h are the MAC payload and header transmission times, t_{plcp} is the transmission times of PLCP header and preamble and t_{ack} is the acknowledgment transmission time.

In TDuCSMA bandwidth reservation is performed, in a per-node basis, by allocating one or more TFs to contend for channel access in accordance with $EDCA^H$. Therefore to node i can be reserved a bandwidth:

$$G_i = \frac{T^i_H}{TC} G_A, \qquad (2)$$

where G_A is the available bandwidth. Nodes sending QoS-demanding traffic experience very few collisions basically at the boundaries of their T_H, *e.g.,* at the beginning of TF 1, 7 and 10 in the example depicted in Fig. 1. Thus, G_A can be estimated by G_{id} with a tolerance of about 10% as verified in [3], [4], [6].

Reverting (2), it is possible to compute the number of TFs n_i that must be allocated to node i to reserve the bandwidth G_i as follows:

$$n_i = \left\lceil \frac{T_C}{T_f} \frac{G_i}{G_A} \right\rceil = \left\lceil \frac{k \cdot T_f}{T_f} \frac{G_i}{G_A} \right\rceil = \left\lceil k \frac{G_i}{G_A} \right\rceil. \qquad (3)$$

It is worth noting that each node on a multi-hop route can exploit (1) to estimate the available bandwidth G_A and (3) to calculate the number of TFs, whose allocation is required to reserve bandwidth G, independently of the others.

However, (1) can be applied only with constant packet length. As demonstrated in [4] the mean packet length provides itself a good approximation of the statistic and the detailed nature of the distribution has only a second order effect when dealing with reservation in TDuCSMA. Therefore (1) and consequently (2) and (3) can be generalized to work with variable packet length as follows:

$$G_{id} = \frac{RT_P}{AIFS^H + 2t_{plcp} + T_P + t_h + SIFS + t_{ack}}, \quad (4)$$

where $T_P = E[t_p]$ is the mean value of the MAC payload transmission time.

III. Prototyping with Wireless Atheros Driver

The prototype exploits OpenWRT, a Linux distribution for embedded devices, and a router board equipped with miniPCI wireless cards based on Atheros chipset. The wireless Atheros drivers are open-source and implemented by the *ath5K kernel module* within OpenWRT. This module gives the kernel the capability to load and unload the miniPCI wireless card and to control the inner functioning of the chip. It does not implement the IEEE 802.11 MAC layer, since it is implemented both by hardware and *mac80211 kernel module* in OpenWRT.

A. Basic Operations

The *ath5k kernel module* comprises several files, but only some of them are really relevant to the goal of TDuCSMA prototyping:

1) **ath5k.h** includes the libraries needed to interact with the rest of the kernel. These libraries are implemented within the *ath common layer kernel module* which defines functions and constants common to all Atheros drivers. The **ath5k.h** file also contains definitions of constants, such as the contention parameters proper of legacy IEEE 802.11 standard, as well as the definition of functions used to perform basic register I/O on the specific class of Atheros chipset.
2) **base.h** defines the software carrier structure called ath5k_softc. This structure is passed to several functions throughout the module and keeps track of the driver state associated with an instance of a device.
3) **base.c** implements several functions to handle PCI interaction (to probe, unload, suspend and resume) and to load and unload the module from the kernel. These functions are used to define an entry point to the whole driver and to perform the initial setup of the module.
4) **qcu.c** defines several functions among which ath5k_hw_reset_tx_queue() that implements the write operation on the registers of the wireless card.
5) **reg.h** defines the memory addresses where to read and write specific registers on the Atheros chipset, such as the ones storing the contention parameters. Thus, it becomes possible to change the channel access parameters by writing these registers at runtime. This file is highly relevant for the TDuCSMA implementation since it relies on the time-driven switching of the contention parameters as described in Section II-A.

The block scheme of the TDuCSMA code implementation is depicted in Fig. 2. The implementation consists of two files included within the *ath5k kernel module*. The **tducsma.h** contains the definition of the tdu_csma structure. This structure includes the contention parameters in the $EDCA^H$ and $EDCA^l$ sets, the CTR structure configuration and an array

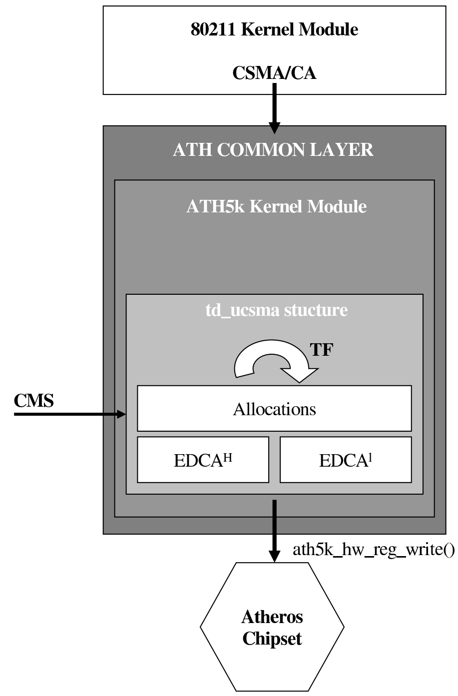

Fig. 2. The block scheme of the TDuCSMA code implementation by Atheros driver within OpenWRT.

representation of the time cycle containing the TF allocations. The elements of this array are equal to 1 if the related TF is allocated to the node and 0 otherwise. Moreover there is a timer to schedule the beginning of the next TF. This timer is implemented by using the Linux 2.6.x kernel built-in high resolution timer *hrtimer* that provides a greater precision than other legacy solutions. The **tducsma.c** contains the implementation of three functions that are called at specific instant during the *ath5k kernel module* execution to implement the TDuCSMA operating principles:

1) tducsma_setup() is called when the *ath5k kernel module* is loaded into the kernel and initializes the tdu_csma structure with default values defined in the **tducsma.h**;
2) tducsma_localization() is periodically called to localize the node within the CTR structure depicted in Fig. 1. It determines the current TF from the system time τ that is synchronized with the other TDuCSMA nodes as follows:
$$\text{TF} = \text{mod}\left(\left\lfloor \frac{\tau(t)}{T_f} \right\rfloor, TC\right) \quad (5)$$
and then schedule the beginning of the next TF after a period of time
$$T_{\text{comp}} = \left\lceil \frac{\tau(t)}{T_f} \right\rceil - \tau(t) \quad (6)$$
to compensate for possible drift in TF beating among nodes.
3) tducsma_callback() is called at the beginning of each TF and schedules itself after a period equal to T_f to implement the periodic TF beating. It writes the contention parameters on the Atheros chipset in accordance with the TF allocation array at each execution. The

parameters in $EDCA^H$ during the TF allocated to it and $EDCA^l$ otherwise. The operation is performed by calling the register write function ath5k_hw_reg_write(), defined in **ath5k.h**, and the register addresses defined in **reg.h**.

The pointer to the tdu_csma structure is added the ath5k_softc structure within **base.h** such that it is possible to access it from any function of the *ath5k kernel module*. As soon as the *ath5k kernel module* is loaded into the kernel, the tducsma_setup() function is called by the function that acts as an entry point, defined in **base.c**, to initialize the tdu_csma structure. Then the *ath5k kernel module* performs a reset operation of the wireless card by calling ath5k_hw_reset_tx_queue(), implemented in the **qcu.c**. Hence the default contention parameters – the ones in $EDCA^l$ – stored in the tdu_csma structure are as well written on the card registers. During this reset operation the function tducsma_localization() is called for the first time in order to localize the node within the CTR structure and to start the periodic TF scheduling. Thus, the function tducsma_callback() is called at the beginning of each TF to write the contention parameters into the registers of the Atheros chipset in accordance with the TF allocations and to schedule the beginning of the next TF.

Note that TDuCSMA code implementation only foresees minor changes to the *ath5k kernel module* to enable the switching of contention parameters in accordance with the CTR structure and TF allocations. Conversely, the basic CSMA/CA operations implemented by the *mac80211 kernel module* are not altered at all and the same is for the underlying hardware. This fact itself confirms that TDuCSMA is IEEE 802.11 standard compliant, as claimed in [3], [4], [5].

B. Time Synchronization

The TDuCSMA relies on a CTR structure to coordinate EDCA parameters switching inside nodes. The synchronization procedure involves two steps: nodes refer to a local clock – the system time τ – that is maintained synchronized with the other nodes, then they exploit this common time in order to deploy the CTR structure and to locate themselves therein. However, the synchronization in TDuCSMA is not so critical [7], because it works at low rate and must be held at TF-level – typically of 1ms – and not at bit-level. Moreover the TF is not a strict slot in which packets must be completely transmitted as in strict TDMA solutions, but it is a period of time during which a node must content for channel access in accordance with specific EDCA parameters. In fact, the channel access is arbitrated by CSMA/CA.

Since the accuracy required for the synchronization is not so critical and the main goal of this first prototype is to prove the feasibility and effectiveness of TDuCSMA given the synchronization, the system clock of each node is maintained synchronized with the others using the Network Time Protocol (NTP) [8] by relying on a single, out of band connection to a local NTP server to reduce latency and jitter, hence the synchronization error.

While NTP fulfills the first step of the synchronization procedure, each node implements the second by tducsma_localization() and tducsma_callback() functions. Note that each node schedules the beginning of the next TF by tducsma_callback() in accordance with its local clock. These clock are prone to drift, hence the TF beating progressively gets un-synchronized among nodes. Therefore, each node has to resynchronize the TF beat after a given number of time cycles. This is achieved by calling again the tducsma_localization() function from tducsma_callback() when needed.

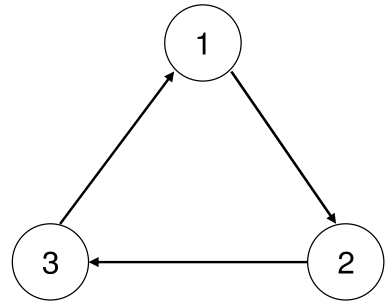

Fig. 3. Testbed diagram; arrows represent long-lasting UDP data flows.

C. Central Management System

A central management system (CMS) has been implemented for this first prototype, to facilitate the remote configuration of TDuCSMA related parameters on all the nodes, *e.g.*, T_f, TC, $EDCA^H$ and $EDCA^l$ sets. The CMS also enables the static allocation of TFs inside nodes, hence allows the user to reserve bandwidth over single routes at run-time. All these parameters are stored, as variables, within the tdu_csma structure in the *ath5k kernel module*. Thus, the virtual file system *sysfs* is exploited to enables their remote configuration by writing the desired values into specific files mapped to the variable themselves.

IV. Wireless Communication Experiments

A. Testbed Setup

The testbed in Fig. 3 is made of three TDuCSMA nodes operating in ad-hoc mode in accordance with IEEE 802.11a on channel 40 at 5.2GHz. Thus, $SIFS = 16\mu s$, $slotTime = 9\mu s$, the PLCP preamble and header are 96 and 24-bit-long respectively. Moreover, the MAC header length is 34 Bytes and ACK length is 14 Bytes. The nodes exchange long-lasting UDP flows whereas line-rate R and MAC payload sizes are changed throughout the experiments. The auto-fall-back is disabled.

In TDuCSMA nodes:

$$AIFS^i = SIFS + AIFSN^i \cdot slotTime \ \forall i = H, l$$

where $AIFSN^H = 2$ and $AIFSN^l = 7$, $CW^H_{min} = CW^H_{max} = 1$ whereas $CW^l_{min} = 31$ and $CW^l_{max} = 1023$. Moreover $TC = 20$ TFs and $T_f = 1ms$.

The experiments are also performed with legacy IEEE 802.11 nodes contending for channel access in accordance with CSMA/CA for the sake of comparison. In these nodes:

$$AIFS = SIFS + 7 \cdot slotTime$$

and the congestion window varies between $CW_{min} = 31$ and $CW_{max} = 1023$.

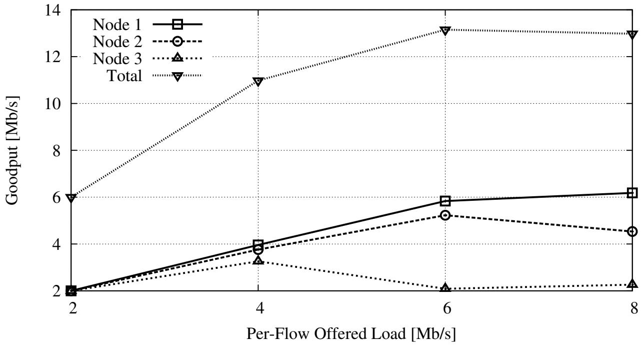

Fig. 4. Goodput at MAC layer as function of per-flow offered load with TDuCSMA; R =18Mb/s and the MAC payload is 1500 Bytes.

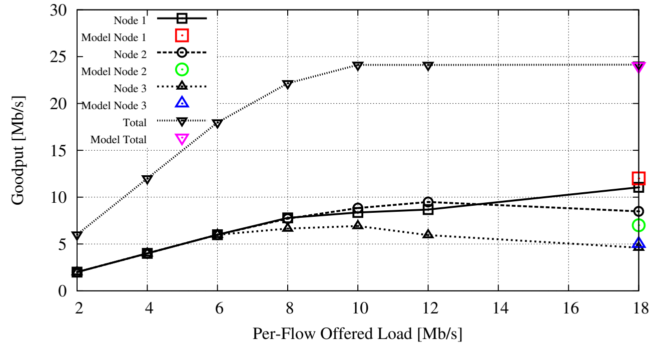

Fig. 5. Goodput at MAC layer as function of per-flow offered load with TDuCSMA; R =36Mb/s and the MAC payload is 1500 Bytes.

B. Results

1) bandwidth management: in the first set of experiments, R =18Mb/s and different T_H are allocated to nodes, in details T_H^1 =10 TFs, T_H^2 =6 TFs and T_H^3 =4 TFs. TDuCSMA nodes send packets with a MAC payload of 1500 Bytes while the data-rate of each flow is increased with a step of 2Mb/s until the network reaches the saturation.

Fig. 4 shows the goodput at MAC layer reached by TDuCSMA nodes. Note that, the total goodput in the graph – calculated as the sum of the goodput reached by each node – coincides with available bandwidth G_A. The goodputs increase fairly among nodes until network saturation takes place; then each node reach a different goodput. The interesting point is that the ratio between the goodput in saturation condition and the available bandwidth well approximate the ratio $T_H^i/TC \; \forall i = 1, 2, 3$. For example G_2/G_A=0.35 and T_H^2/TC=0.3. As a consequence, since the available bandwidth G_A can be estimated starting from (1), as described in Section II-B, the equations (2) and (3) represent an viable model for bandwidth reservation on real TDuCSMA networks.

A second set of experiments is run to prove this latter claim. In these experiments, R =36Mb/s and TDuCSMA nodes send packets with a MAC payload of 1500 Bytes. Given the setup and applying (1), the estimation of the available bandwidth $G_A = 0.9 \cdot G_{id} \approx$ 24Mb/s . A share of G_A is reserved to each TDuCSMA node, by means of (2) and (3), such that 12Mb/s were reserved to Node 1, 7Mb/s to Node 2 and 5Mb/s to Node 3. The data-rate of each flow is increased with a step of 2Mb/s until the network reaches the saturation.

Fig. 5 shows the goodput at MAC layer reached by TDuCSMA nodes. The results follow the same trend of the previous experiments and, above all, they confirm the effectiveness of the reservation model. In fact, the maximum deviation between the reservation value (color dots in the graph) and the goodput achieved is less than 6% with respect to the available bandwidth. The maximum deviation occurs for Node 2. Moreover it is worth noting that, when the same experiments is run with CSMA/CA nodes the available bandwidth decreases from 24 to 20Mb/s in saturation condition. This is due to the lower collision probability assured by TDuCSMA.

2) effect of packet length: a third set of experiments is run to assess the effect of packet length on the bandwidth management capability of TDuCSMA. In these experiments R =18Mb/s, Node 1 and 3 send packets with a MAC payload of 1500 Bytes whereas Node 2 sends packets with a MAC payload of 500 Bytes. A share of G_A is reserved to each TDuCSMA node, such that 7Mb/s were reserved to Node 1 and 3Mb/s to Node 3. After these two reservations the number of un-allocated TFs is equal to 6 and all of them are allocated to Node 2. Since Node 2 sends short packets the reservation is for 3Mb/s. Note that if Node 2 sends packets with a MAC payload equal to 1500 Bytes the reservation would be about 4Mb/s. This means that it is using its T_H with poor efficiency. The data-rate of each flow is increased with a step of 2Mb/s until the network reaches the saturation.

Fig. 6 shows that each node exploits its reservation with TDuCSMA while the transmission inefficiency of Node 2 does not affect the performance of the other nodes. In fact, the maximum deviation between the reservation value (color dots in the graph) and the goodput achieved is less than 2% with respect to the available bandwidth. The maximum deviation occurs for Node 1. On the contrary, Fig. 7 shows that the transmission inefficiency of Node 2 affects also the performance of nodes sending long packets with CSMA/CA. The total goodput decreases of about 1.3Mb/s from 12.9Mb/s to 11.6Mb/s. Previous experiments showed that the gain in the total goodput of TDuCSMA over CSMA/CA is due to collision probability when all nodes send packets with the same length and this gain increases with the line-rate. However when $R \leq$ 18Mb/s this gain is very low. Therefore the above reduction of about 10% is due to the effect of short packets.

3) effect of misbehaving nodes: the fourth set of experiments evaluate how TDuCSMA reacts to misbehaving nodes, *i.e.*, nodes transmitting more traffic than reservation. In these experiments R =18Mb/s and all nodes send packets with MAC payload equal to 1500 Bytes. Given the setup and applying (1), the estimation of the available bandwidth $G_A = 0.9 \cdot G_{id} \approx$ 13Mb/s . A share of G_A is reserved to each TDuCSMA node, by means of (2) and (3), such that 7Mb/s is reserved to Node 1, 4Mb/s to Node 2 and 3Mb/s to Node 3. Moreover, Node 2 sends with a data-rate of 8Mb/s –

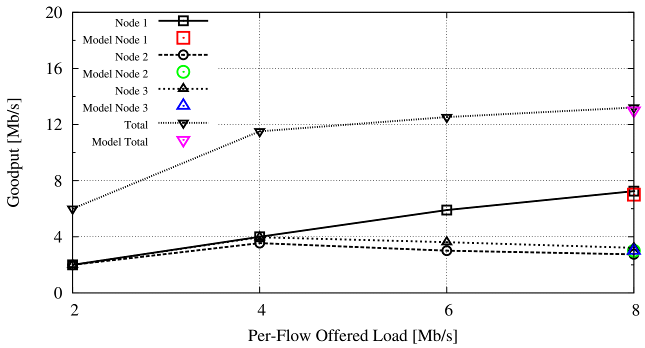

Fig. 6. Goodput at MAC layer as function of per-flow offered load with TDuCSMA; R =18Mb/s and the MAC payload of Node 2 is 500 Bytes.

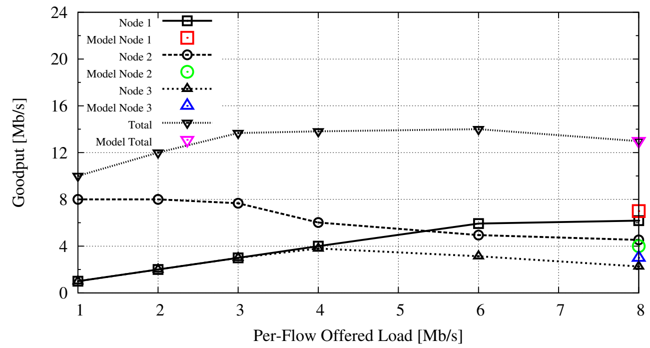

Fig. 8. Goodput at MAC layer as function of per-flow offered load with TDuCSMA; R=18Mb/s and the data-rate of Node 2 is twice the reservation.

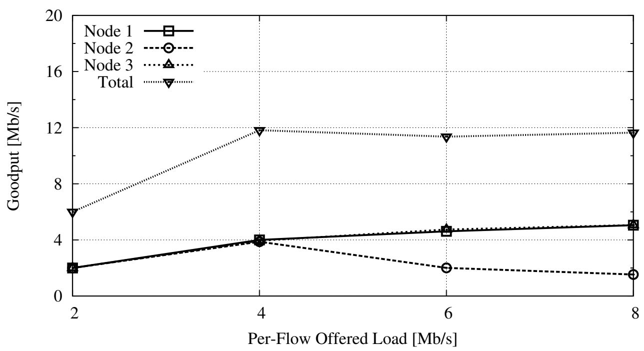

Fig. 7. Goodput at MAC layer as function of per-flow offered load with CSMA/CA; R =18Mb/s and the MAC payload of Node 2 is 500 Bytes.

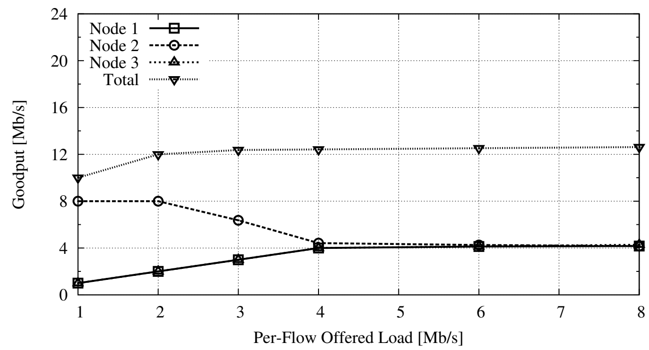

Fig. 9. Goodput at MAC layer as function of per-flow offered load with CSMA/CA; R=18Mb/s and the data-rate of Node 2 is twice the reservation.

twice the reservation – while the data-rate of the other nodes is increased with a step of 1Mb/s.

Fig. 8 shows that TDuCSMA gets through all the traffic from Node 2 until the offered load is less than G_A. This means that TDuCSMA is *adaptive* and it intrinsically allows bandwidth re-use. This is a great advantage with respect to traditional TDMA solutions where re-use must be implemented with specific functions and it represents indeed a cost in term of complexity. Fig. 8 also shows that TDuCSMA behaves fairly by *policing* exceeding traffic generated by Node 2, as the other nodes increase the data-rate, such that all nodes achieve a goodput equal to the reservations. Conversely CSMA/CA cannot carry out any policy mechanisms, as depicted in Fig. 9. All nodes reach the same goodput in accordance with the fair-share behavior of CSMA/CA.

These last results also show a great advantage of TDuCSMA over CSMA/CA that is the possibility to handle unbalanced traffic flows – flows with different data-rate – on the same network due to its bandwidth management capabilities.

V. Conclusion and Future Work

This paper has presented the work of prototyping TDuCSMA by modification of the wireless Atheros drivers. The achievement of the prototype and the results of the real communication experiments have demonstrated the feasibility of this novel coordination function, its capabilities in bandwidth management and QoS provisioning and the effectiveness of the bandwidth reservation model as an analytical tool for bandwidth management in real networks. Future work will be devoted to improve this first prototype by implementing a distributed leaderless synchronization solution [7] and a signaling architecture [6] for distributed and dynamic bandwidth management.

References

[1] M. Baldi, R. Giacomelli, and G. Marchetto, "Time-Driven Access and Forwarding for Industrial Wireless Multi-hop Networks," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 2, pp. 99–112, May 2009.

[2] E. Carlson, C. Prehofer, C. Bettstetter, H. Karl, and A. Wolisz, "A Distributed End-to-End Reservation Protocol for IEEE 802.11-Based Wireless Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2018–2027, Nov. 2006.

[3] A. Vesco and R. Scopigno, "Time-division Access Priority in CSMA/CA," in *Proc. of IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2009)*, Sep. 2009, pp. 2162–2166.

[4] R. Scopigno and A. Vesco, "A Distributed Bandwidth Management Scheme for Multi-Hop Wireless Access Networks," in *Proc. of Wireless Communications and Mobile Computing Conference (IWCMC 2011)*, Jul. 2011, pp. 534–539.

[5] A. Vesco and R. Scopigno, "Advances on Time-Division Unbalanced Carrier Sense Multiple Access," in *Proc. of IEEE Conference on Computer Communications and Networks (ICCCN 2010)*, Aug. 2011, pp. 1–6.

[6] A. Vesco and Scopigno, "A weakly synchronous and distributed coordination function for QoS management in CSMA/CA wireless access networks," in *Developments in Wireless Network Prototyping, Design and Deployment: Future Generation*, M.A. Matin, Ed. IGI Global, 2011.

[7] A. Vesco, F. Abrate, and R. Scopigno, "Convergence and Performance Analysis of Leaderless Synchronization in Wi-Fi Networks," in *Proc. of ACM Workshop PM2HW2N 2011*, Nov. 2011, pp. 49–58.

[8] D. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.