Context-Aware Packet Switching in Ad Hoc Networks

J.J. Garcia-Luna-Aceves[†‡] Marc Mosko[†] Ignacio Solis[†] Rebecca Braynard[†] Rumi Ghosh[†]
[‡] Computer Engineering Department
University of California at Santa Cruz
Santa Cruz, CA 95064
[†] Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

***Abstract*—We present the design and performance of a new approach to packet switching for MANETs, which we call Context Aware Protocol Engines (CAPE). With CAPE, nodes disseminate information in the network by means of context-aware packet switching that enables the statistical multiplexing of bandwidth, processing and storage resources using integrated signaling covering channel access, routing and other functions, to share and store the context within which information is disseminated. Data packet headers consist of simple pointers to their context, and elections and opportunistic reservations integrated with routing are used to attain high throughput and low channel-access delay.**

I. Introduction

The protocols and architectures used in mobile ad hoc networks (MANET) today [1]–[3] still reflect the severe memory and processing constraints imposed on computing equipment dedicated to communication tasks 40 years ago [4]–[6]. Because of such constraints, the protocols used in the ARPANET had to be organized into a stack in which most protocols were decoupled from the physical medium, each protocol layer operated independently of other, and processing and storage "inside" the network was kept to a minimum. The resulting packet switching architecture was based on "in-band, in-packet signaling" aimed at minimizing the use of in-network processing and storage resources while maximizing the utilization of communication links. In this approach, routers (packet switches) maintain the minimum needed to forward packets (e.g., a next hop); each packet contains a header and a payload, with the header carrying all the control information needed for each protocol layer of the protocol stack; each packet (datagram) is switched independently of others or the intent or type of its payload, and all processing and storage of content occurs at the edges of the network (i.e., at the hosts).

Given the success of the IP Internet, datagram switching based on protocol stacks was arguably the most sensible approach to packet switching at the time the ARPANET was created. Furthermore, it is still adequate for wired segments of the Internet in which over-provisioning of links is feasible. However, "stacked datagrams" it is not the right packet switching approach for today's MANETs. MANETs are very different than a wired network due to node mobility, the characteristics of radio channels, and the relative scarcity of bandwidth (compared to fiber). The in-network processing and storage power available even in small mobile nodes (e.g., PDAs, cellular phones) today are orders of magnitude larger than what was available inside a network more than 40 years ago [6]. Furthermore, while there were compeling cost reasons for a division of labor between hosts and routers (switches) 40 years ago, the wireless portions of the Internet need to be ubiquitous and invisible, each node of a MANET must act as a host and a router, and providing the user with the service or content she wants is far more important than trying to attain high link utilization. Interestingly, as our review in Section II of prior work on new approaches to packet swithcing in MANET indicates, all prior work has adhered to at least some of the original approach to packet switching introduced in the ARPANET.

In this paper, we show by example that context-aware statistical multiplexing of network resources is far more effective (in some cases by orders of magnitude) than implementing datagram switching using protocol stacks. Section III introduces our approach, which we call Context Aware Protocol Engines (CAPE), as an instantiation of context-aware packet switching for MANETs. The approach we advocate in CAPE is the exact opposite approach than today's protocol architectures. CAPE is based on nodes storing the entire context within which packets are to be switched, and each packet contains its payload and a pointer to bind it to the stored context. The signaling used in CAPE is "in-band, off-packet integrated signaling," in that the same links are used to communicate control and data, but signaling and control information used to establish how a given data packet is to be processed is disseminated ahead of the transmissions containing user data, and a single protocol is used to establish the context for the dissemination of packets, and such context includes all the control information needed to attain integrated scheduling, routing and congestion control decisions. A key advantage of integrating the signaling used for scheduling and routing is that the schedules obtained to access the shared channel take into account the constraints imposed by flows that must hop over multiple relays, as well as network-level knowledge of which neighbors of a node should be receiving the transmission. The protocol used for signaling and channel access in CAPE is the

Context Aware Scheduled Access (CASA) protocol. Channel access with CASA is based on a combination of distributed fair elections to determine which nodes are allowed to use or bid for unreserved time slots, together with reservation mechanisms to provide channel-access time guarantees to those nodes that have succeeding accessing the channel and must persist using the same time slots.

The implications on network efficiency resulting from using "stacked datagrams" in MANETs is most evident in the low efficiency with which bandwidth is shared and the large overhead incurred with context-free packet switching of information. Accordingly, while CAPE spans all the layers of a traditional protocol stack, this paper focuses primarily on the benefits obtained with context-aware channel access and integrated signaling. Section V compares the performance of CASA and the overhead improvements attributed to integrated signaling against the use of traditional protocol stacks based on IEEE 802.11 DCF. The results clearly show that the use of a contention-based channel access discipline that, in essence, attempts to emulate "Ethernets in the sky," is simply not applicable to MANETs with voice and data traffic. The context-aware channel access used in CAPE provides a solid foundation for the support of multiple media in MANETs. The overhead incurred with datagram switching implemented with layer-independent signaling can be substantial when either the payload per packet is small or the duration of an end-to-end flow is relatively long (involving many packets). Our results indicate that the integrated signaling in CAPE provides reduced overhead even when the signaling asociated with route maintenance and end-to-end transport is not considered.

II. Related Work

There has been considerable work in the past attempting to improve the efficiency and scaling of ad hoc networks. We organize this work into: (a) protocol-header compaction or compression, (b) cross-layer optimizations, and (c) new protocol architectures.

Protocol-header compaction proposals focus on reducing the overhead of the headers required in specific protocols, without incurring any information loss, while compression proposals allow some loss of information. Header compression or compaction approaches are based on at least one of the following observations: (a) many header fields of packets in the packet stream of an end-to-end session are the same (e.g., source and destination address, ports, version, protocol, flow label, hop limit), (b) nodes can use local identifiers instead of globally unique identifiers provided that they maintain a mapping for them, and (c) protocol headers unnecessarily carry all the fields that the processing of a packet may require. Most of the attention has been given to the overhead of TCP, UDP, and IP in ad hoc networks [7]–[10]. What is most striking about all the approaches to date on compaction and compression is that they all assume a layered protocol architecture based on in-band in-packet signaling in which one layer encapsulates the higher layer and all protocol headers are included in each packet. The goal is simply to attempt to reduce overhead of specific protocols defined within that architecture. Consequently, the benefits of cross-layer interaction are not fully exploited.

Because the characteristics of the physical layer impact the performance of the entire protocol stack in an ad hoc network, much work has recently been done attempting to integrate the operation of multiple layers to make them more efficient. To date, cross-layer optimization schemes have focused on exploiting advances at the physical layer and have addressed: (a) using multiple channels at the link and network layers (e.g. [11], (b) taking advantage of more sophisticated receivers (e.g. [12]), and (c) taking advantage of information processing by the relaying nodes [13]–[16]

The importance of these prior results is that taking advantage of multi-packet reception (MPR) for channel access benefits from dynamic approaches to channel division (in time, frequency, and space), and requires the scheduling of concurrent transmissions around receivers based on their characteristics and the channel state at the receivers. Hence, truly scalable protocol architectures for ad hoc networks need to consider the use of scheduled channel access.

Very few new architectures have been proposed to improve the performance of ad hoc networks. The majority of the architectures proposed for ad hoc networks in the past have focused on organizing nodes into clusters (identified either by a cluster identifier or a node identifier) or into a connected backbone that links all nodes to reduce signaling overhead. Recently, however, Ramanathan [17] proposed an architecture based on three layers: a relay-oriented physical layer, a path access control (PAC) layer, and a collaborative transport layer. While Ramanathan's proposal captures many of our goals in CAPE, it has some limitations. The PAC layer is subject to unfair access to resources much more so than today's 802.11 DCF is, because the right to use resources at the physical layer is entirely based on a reservation handshake carried without information about the context within which flows are being transmitted. In addition, while cooperation among nodes is important, it may not be possible for the transmitters to have accurate channel state information when nodes move fast.

III. Context Aware Protocol Engines (CAPE)

Contrary to the case of wired networks, scheduling, routing, congestion control and retransmission control are very much interrelated in a MANET. A transmission schedule, in effect, defines a "link" between a transmitter and a receiver. A route established and used dictates the maintenance of links and the decay of others, and therefore impacts the schedules. Lastly, the routes established determine the congestion to which links are subjected, and changing routes or the allocation of traffic to such routes impact congestion. Therefore, establishing the context within which resources are shared and information is exchanged needs to take place together with channel access, and channel access needs to be performed jointly with the other network control functions.

Accordingly, in CAPE, we substitute the traditional protocol stack with: (a) A context database storing the context within which user information is disseminated and network resources are shared; (b) the Context Aware Scheduled Access (CASA)

protocol, which is used to access the shared channel and disseminate all context and user information; and (c) a set of network-control algorithms.

The information in the context database includes information about the flows competing for shared bandwidth, the nodes capable of causing interference around receivers, link characteristics, node positions (if available), transmission schedules and routes, other characteristics of the environment that may also help define the context, and state information used by network control algorithms.

CASA integrates the signaling required for channel access, routing, congestion control, and retransmission control. Instead of having a medium access control (MAC) protocol, a unicast routing protocol, a multicast routing protocol, and a congestion control protocol each carry its own signaling independently of the others, nodes exchange their context with one another using CASA, and such context includes all the control information needed to attain integrated scheduling, routing and congestion control decisions. Although CASA supports the signaling for multiple functions, the algorithms used to implement different network control functions are *not* integrated into a single algorithm, given that such an optimization problem would be much too complex.

Environment context packets contain information about local neighbors and routes. They occur periodically and maintain routing tables and the neighborhood information required by the scheduler. Flow context packets on the other hand relate information about a specific data flow. They are in charge of setup, update and teardown of the context information needed to forward the actual data.

CASA is a time-slotted MAC layer that uses a slot header and a per-packet packet header in aggregated slots. Both headers are 64 bits. The slot header identifies who the sender is, what version of CASA is used, control flags, and the number of packets in the slot. The packet header contains the identifier of the context within which the packet should be processed, the version of the context identifier, the size of the packet. This information determines the format of the context-dependent header (CDH) and how to process the packet. An example of data that might be part of the context-dependent header is a *link sequence number*.

Data flows are established by first setting up a context using context setup packets. In CAPE, a source identifies a flow by a globally unique Flow ID (FID), and the FID is propagated through the network as part of the the connection setup. The FID takes a role akin to an IP socket descriptor: it uniquely identifies a flow between endpoints. Once a pair of neighbors along a flow path agrees on the setup, the nodes switch the flow by using the context id. However, because nodes know the FID associated with specific peer connections, nodes may multiplex a flow over multipaths. The cost of path repairs in CAPE is minimized because nodes can locally re-route a flow using the FID.

Setup packets are used to build the context. They allow different types of data flows to be set up. Figure 1 shows a graphical representation of a setup packet. The next hop determines the node that should receive the packet next. The destination is the destination of the flow. Flow ID is the globaly

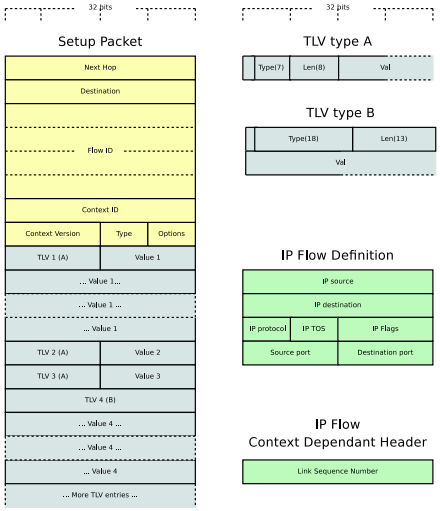

Fig. 1. CASA packets and fields

unique identifier for this flow. Version determines the version of the context. Type determines what kind of flow this is, an example would be an encapsulated IP flow, or a CAPE native flow. The setup packet also contains a set of TLV (type-length-value) entries that define the flow and create the context. There are two types of TLV entries differentiated by the first bit. Type A uses 7 bits for Type and 8 bits for length. This will be used for the most common options. Type B uses 18 bits for type and 13 bits for length, this will handle the extended options. One of the TLV entries present in most setup packets will be the Flow Definition (IP Flow definition can be seen in figure 1).

The context defines how to interpret the context-dependent header, and includes information that needs to be processed specifically for a given flow. In the case of an IP flow, the header includes a field called *Link Sequence Number*. This is a sequence number for the packets of this flow over the specific link in which they are sent. It is used for link retransmissions and flow control.

IV. Channel Access

CASA provides access to a shared channel by means of elections and reservations based on the context information exchanged among nodes. Nodes implement a distributed election algorithm to select which node is allowed to use time slots that have not been reserved using the context they share to run the election, without the need to configure anything in the schedule other than the number of time slots used in each frame. Reservations are used to provide channel-access time guarantees to those nodes that have succeeded accessing the channel and must persist using the same time slots oer time. To calculate schedules and propagate reservations, nodes regularly broadcast their context. The context needed for channel access consists of neighborhood and reservation information. Nodes also utilize an in-band time synchronization protocol that does not rely on external time sources such as GPS. The remainder of this section details the components of CASA: neighbor maintenance, slot elections, reservations, and in-band time synchronization.

TABLE I
NOTATION

ND_j^i	Hop-count distance at node i to neighbor j
NA_j^i	Age since last update of ND_j^i
T_{nbr}	Timeout for neighbor updates (default 1S)
T_1	Reservation timeout at reserving node
T_2	Reservation timeout at non-reserving node
M	Number of slots in a frame
R_{max}	Max reservations per frame
R_{new}	Max new reservations per node per frame
RH_i	Number of reservations held by node i (default maximum 80% of M)
RN_i	New reservations in the current frame by node i (default 1% of M)

Neighbor maintenance is used to populate an N-hop neighbor table, which is the basis for the dynamic election of time slots that have not been reserved. A neighbor update is the pair $\{id, distance\}$, where id is the node ID of a neighbor and $distance$ is the hop count from the node. When sending context, all neighbors of distance less than N (for an N-hop neighborhood) are sent at once in one long array of neighbor updates, though an incremental approach is also possible. When sending the mini-context, a node fills in the nbr_id with any distance 1 neighbor. We use a round-robin of all distance 1 neighbors from slot-to-slot.

A neighbor j of node i is declared a valid N-hop neighbor if and only if ND_j^i exists and $ND_j^i \leq N$ and $currentTime - NA_j^i \leq T_{nbr}$. For the remainder, when we say a node j is an N-hop neighbor of i, we mean this definition.

Upon receiving a neighbor update, a node updates the distance to the neighbor. A neighbor update may be received three possible ways: inferred from a slot header, inferred from the mini-context, or from a neighbor update. From a slot header, node i assumes a distance d of 0. From a mini-context, a node assumes a distance of 1. From a neighbor update, a node uses the stated neighbor distance. Upon receiving a neighbor update, if it is a new neighbor, node i creates $ND_j^i \leftarrow d + 1$ and $NA_j^i \leftarrow currentTime$. If the neighbor entry already exists, NA_j^i is updated and ND_j^i is updated only if the distance decreased.

A node executes a cleanup algorithm at the beginning of every frame. This removes any old neighbor entries and pushes back (i.e. increments the distance) any timed out entries. For each neighbor j of i, if $currentTime - NA_j^i > T_{nbr}$, then ND_j^i is incremented. If $ND_j^i > N$, then the neighbor entry is removed.

1) Time-Slot Scheduling: Previous dynamic scheduling protocols [18]–[20] offer high channel utilization even at high load, but still have problems with real-time data traffic, because their randomized slot assignments do not provide bounds on channel access times. Our approach maintains high utilization without sacrificing real-time data flows.

A major difference in the time-slot elections in CASA compared to previous election-based scheduling schemes is that prior work has focused on running an election for each individual time slot, while elections in CASA run for an entire frame. CASA's election algorithm is based on generating a pseudo-random permutation of time slots for each node. These permutations are then compared to determine which node wins the right to transmit in each time slot. Although the permutations are frame-length, the permutations and election algorithm may be run in amortized time over a frame. For each frame number t, define $K(t)$ to be some random number called the frame key. We assume that all nodes know the common frame key $K(t)$. The frame key is used as the random number seed used to generate the random permutation vectors.

Algorithm 1:
RUN ELECTION($P[node][slot], W[node][slot], N, M$)
(1) $S[1, \ldots, M]$: winning ids, initialized to $N + 1$
(2) $R[1, \ldots, M]$: the rank ordering of node winning slot
(3) $V[1, \ldots, M]$: the weight of node winning slot
(4) **for** $t = 1$ **to** M
(5) **for** $n = 1$ **to** N
(6) **if** $S[t] = N + 1$ or ($R[t] = t$ and $W[n, t] < V[t]$)
(7) $S[t] \leftarrow n$
(8) $R[t] \leftarrow t$
(9) $V[t] \leftarrow W[n, t]$
(10) **return** S

Algorithm 1 shows how to determine the schedule of winning node IDs given the set of random permutation vectors in the matrix $P[node][slot]$ and the set of node weights for tie-breaking. After one generates the random permutation vectors, Alg. 1 scans them in time-slot order such that the node with the minimum rank for a time slot winds the slot. The weights could be random numbers or they could be a deterministic value based on node ID. If the weights are independent of the slot time and have the same sort order as the node identifiers, the second clause of line 6 and vectors R and V could be removed.

When used with reservations, the reservation for a slot always takes precedence over the election results. Each node computes the election results exactly as in Alg. 1, but makes a final check to see if the slot is reserved. If the slot is reserved by the node itself, it may transmit in the slot without winning the election. If the slot is reserved by different node, then the local node cannot transmit in the slot, even if it won the election.

2) Reservations: The present reservation scheme is designed to support voice calls, and as such is a "hold until done" strategy for keeping a reservation until it is no longer needed. When a node has data to send, it may reserve any slot for which it wins the election, while below the reservation limits. CASA limits a node in the rate of slot reservations and in the maximum number of reservable slots. Nodes can keep their slot reservation as long as they have qualifying data (described below). When a neighbor receives a slot reservation, it propagates the reservation information over the N-hop contention area so the participating nodes can use it in the distributed election. If a reservation is not refreshed, neighbors timeout stale reservations using a soft-state approach.

A node may reserve a time slot if several conditions are met. The total reservations held by node i must be less than the global maximum: $RH_i < R_{max}$. The number of new reservations per-frame must also be less than the allowable: $RN_i < R_{new}$. The type of traffic to be sent in the slot must

also qualify. If there is at least one voice packet scheduled or there are at least two queued data packets, the traffic qualifies for a reservation. We found through experimentation that this definition of qualifying traffic yields good overall performance, but other strategies could be employed. To reserve a slot, a node sets the 'R' flag in the slot header $flags$ field. The node will continue to set the 'R' flag in the time slot as long as it has qualifying traffic to send in that slot. If at least T_1 seconds pass without qualifying traffic, the node releases its reservation and can no longer use the slot without winning it again in an election.

When a node receives a slot header with the 'R' flag, it will record the tuple {*res_id, res_slot, res_seen, res_sent*}, where res_id is the node identifier from the slot header, res_slot is the slot number, res_seen is initialized to the current time, and res_sent is initialized to 0. As part of a context control message, every node will transmit a list of the known reservations that are no older than T_2. The reservations are ordered by res_sent such that the newest (initialized to 0) and least-recently sent reservations go first. Periodically, such as on slot 0 of each frame, a node runs a maintenance routine that purges the reservation cache of any expired reservations. An expired reservation has an age greater than T_2.

A node may hear about multiple reservations for the same time slot. However, if the node is also one of the reserving nodes, it must perform contention resolution, such that no two nodes in a contention area hold the reservation. We use a random tie-break based on the hash of the node ID and slot number.

3) Time Synchronization: We chose the CSMNS protocol [21] for time synchronization in CASA, because of its simplicity. The basis of CSMNS is the use of a proportional controller to drive the clock at each node to a common time. Let node i have a real-time clock with drift β_i and initial phase offset α_i, then the drift time, $t_i^{(d)}$, at node i is

$$t_i^{(d)}(t) = \beta_i \cdot t + \alpha_i. \quad (1)$$

Each node has a control factor s_i such that the synchronized time, $t_i^{(s)}$, at node i is $t_i^{(s)} = s_i \cdot t_i^{(d)}$. In each slot header, a node transmits its timestamp $t_i^{(s)}$ as a 64-bit integer. To aid convergence, we also adopt the rule that if a node has never received a timestamp before, it adjusts its offset to equal the received timestamp, such that with $s_i = 1$, the two clocks are perfectly synchronized.

The control factor is maintained by the synchronization protocol through a proportional control equation. The initial factor is $s_i = 1$. Each time node i receives a time synchronization packet from a neighbor, it updates s_i. Let node i receive such a packet with the timestamp $t_j^{(rx)}$ in the packet header from node j. Node i updates s_i as $s_i = s_i + K * \frac{t_j^{(rx)} - t_i^{(s)}}{t_i^{(s)}}$. The term K is the proportional factor, and typically takes a value in the range 0.3 to 0.8.

V. Performance Evaluation

In this section we present our evaluation of CAPE and compare its performance to 802.11e using the OLSR [22] routing protocol in the Qualnet simulator [23] using the parameters shown in Table II. The simulations are done for a 50 node static mesh environment and evaluate how CAPE handles a mix of web and voice traffic. We consider the metrics of delivery, latency, and HTTP bytes delivered. We will show that CAPE has comparable CBR delivery to 802.11e with few CBR flows, but CAPE is over 3x better with many CBR flows and that CAPE's scheduling maintains low latency that is over an order of magnitude less delay than 802.11e while delivering more elastic TCP traffic. The graphs show the average of 5 independent simulation runs and the 95% confidence interval (assumed Normal). We show that CAPE outperforms 802.11e in a variety of topologies and workloads.

TABLE II
Simulation Parameters

Terrain	2500m x 1000m
PHY	802.11a at 12 Mbps
802.11e	AC3 voice, AC0 TCP
Radio model	Statistical propagation, two-ray ground pathloss, constant shadowing, -111 dB propagation limit
Voice data	56 bytes CBR (17.6 kbps codec), 30s exp. turnaround
Bulk data	50 HTTP sessions [24]
CASA frame	400 slots at 0.5 msec/slot (650 byte MTU)
Contention area	4 hops
Guard interval	10 μsec
Reservations	300 out of 400 slots reservable
Res timeouts	1601 msec local, 2001 msec neighbor
Res rate	at most 4 new reservations/node/frame

CASA has few collisions due to being a contention-free MAC protocol, but packets can still be dropped due to fading channels, errors due to additive noise or collisions in the infrequent case of stale context. All CASA experiments are run with an acknowledgement schemed called "TcpAcks". This scheme only acknowledges and retransmits TCP traffic; all CBR and routing UDP traffic is strictly best-effort. TcpAck utilizes a basic acknowledgement scheme that transmits an ACK for each TCP packet. Packets are retransmitted after 50ms of no ACK is received by the sender. Packets are retransmitted a maximum of three times.

Fig. 2 shows the delivery ratio of CBR packets. The delivery ratio is the total number of in-order datagrams received by a destination divided by the total number of datagrams sent by all senders. The 802.11e plots use standard 802.11 RTS/CTS/ACK with retransmissions. The CAPE plot using CASA does not ACK or retransmit any data. Despite this lack of link ARQ, CAPE has a fairly flat delivery ratio over all loads. CASA and 802.11 have statstically equivalent delivery ratios when there are only 5 CBR flows. Beyond 5 CBR flows, the 802.11 delivery ratio drops quickly below the CASA performance.

Fig. 3 shows the end-to-end CBR latency. CASA has a significantly lower delay than 802.11e in all cases. The 802.11e delay ranges from 1S to 6S. The CASA delay is between 40 msec to 60 msec for up to 25 CBR flows and then up to 160 msec for 50 CBR flows. This means that even with each node having a 16 kbps conversation with another node and each node browsing we pages, CASA maintains a high CBR delivery ratio and low CBR delay, whereas 802.11e

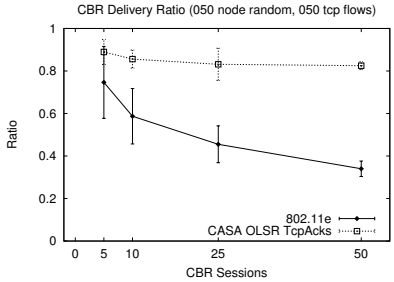

Fig. 2. Delivery ratio 50 HTTP

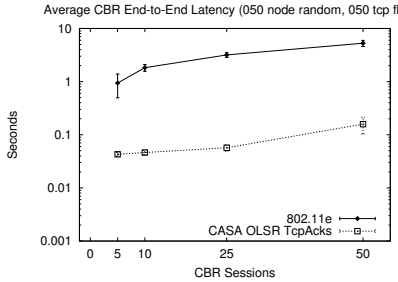

Fig. 3. CBR Latency 50 HTTP

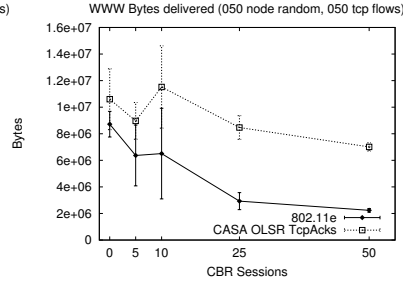

Fig. 4. HTTP Bytes 50 HTTP

breaks down after about 5 CBR flows.

Fig. 4 shows the total HTTP bytes delivered over TCP. In these scenarios, CASA uses the TcpAck mechanism to use link ARQ on TCP packets. 802.11e and CASA delivery statistically equivalent bytes at low load (up to 10 CBR sessions), but at 25 and 50 sessions, CASA delivers about 3x the bytes of 802.11e.

Although not presented here, all experiments were run with both in-band time synchronization and perfect timing. In our experiments, the drift is chosen uniformly between ± 25 ppm, which is similar to the 802.11 requirements for clocks. The initial phase offset is chosen in the range [0, 100] μsec, similar to Rentel [21], with $K = 0.5$. The results show that the uncontrolled drift clock has a mean absolute error of $3,656\mu$sec with a standard deviation of 2148 μsec. Assuming a Normal distribution, the 99% confidence interval ($\pm 2.576\sigma$) spread of clocks would be over 11 msec. For the controlled clocks, the mean absolute error is between 1.205 to 5.669μsec. More importantly, the standard deviation of the error is well controlled between 0.825 to 9.431μsec. This would give a worst-case spread over a 48μsec interval for the 9.431μsec case. The majority of the deviations are close to 2μsec, which has a 99% confidence interval spread of 10.3μsec, which works well with the 10 μsec guard interval.

VI. Conclusions and Future Work

We introduced CAPE as an example of context-aware packet switching in MANETs. CAPE is based on nodes storing the entire context within which packets are to be switched, and having each data packet consisting only of its payload and a pointer to bind it to the stored context. All signaling needed in CAPE is exchanged by means of a single protocol, the Context Aware Scheduled Access (CASA) protocol. We have shown that CASA, the protocol used in CAPE for integrated signaling and channel access, greatly improves channel utilization under heavy loads. It also performs very well in cross traffic scenarios, allowing the multiple flows to cooperate and share the channel. To make CAPE a reality, future work that must be undertaken includes the integration of routing with the scheduling and reservations described for channel access in CASA, and the introduction of hop-by-hop congestion and retransmission control.

References

[1] R.E. Kahn, S.A. Gronemeyer, J. Burchfiel, and R.C. Kunzelman, "Advances in packet ratio technology," *Proceedings of the IEEE*, vol. 66, no. 11, pp. 1468–1496, Nov. 1978.

[2] B.M. Leiner, D.L. Nielson, and F.A. Tobagi, "Issues in packet radio network design," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 6–20, Jan. 1987.

[3] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," *Proceedings of the IEEE*, vol. 75, no. 1, pp. 21–32, Jan. 1987.

[4] R. Kahn, H. Frank, and L. Kleinrock, "Computer communication network design: Experience with theory and practice," *Networks*, vol. 2, no. 2, pp. 135 – 166, 1972.

[5] L.G. Roberts and B.D. Wessler, *Computer-communication networks*, chapter The ARPA Computer Network, Prentice-Hall, 1973.

[6] S. Ornstein, F. Heart, W. Crowther, S. B. Russell, H. K. Rising, and A. Michel, "The terminal IMP for the ARPA computer network," in *Proc. AFIPS Spring Joint Computer Conference*, 1972, pp. 243 – 254.

[7] Mikael Degermark, Mathias Engan, Björn Nordgren, and Stephen Pink, "Low-loss tcp/ip header compression for wireless networks," in *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, 1996, pp. 1–14.

[8] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links," RFC 2508 (Proposed Standard), Feb. 1999.

[9] M.A. Spohn and J.J. Garcia-Luna-Aceves, "Exploiting relative addressing and virual overlays in ad hoc neworks with bandwidth and processing constraints," in *Proc. ICWN*, Jun. 2003.

[10] I. Solis, K. Obraczka, and J. Marcos, "FLIP: a flexible protocol for efficient communication between heterogeneous devices," in *Proc. IEEE ISCC*. IEEE Comput.Soc, July 2001, pp. 100–6, IEEE Comput. Soc.

[11] P. Kyasanur, X. Yang, and N. Vaidya, "Mesh networking protocols to exploit physical layer capabilities," in *Proc. IEE WiMesh*, Sep. 2005.

[12] J. Park and M. Gerla, "Mimoman: A mimo mac protocol for ad hoc networks," in *Proc. AdHocNow*, Oct. 2005.

[13] S. Toumpis and A.J. Goldsmith, "Capacity regions for wireless ad hoc networks," *Wireless Communications, IEEE Transactions on*, vol. 2, no. 4, pp. 736–748, July 2003.

[14] R. M. de Moraes, H. R. Sadjadpour, and J. J. Garcia-Luna-Aceves, "Many-to-many communication: A new approach for collaboration in manets," *IEEE INFOCOM*, pp. 1829–1837, May 2007.

[15] A. Ozgur, O. Leveque, and D. Tse, "Hierarchical cooperation achieves linear capacity scaling in ad hoc networks," *IEEE INFOCOM*, pp. 382–390, May 2007.

[16] J. J. Garcia-Luna-Aceves, Hamid R. Sadjadpour, and Zheng Wang, "Challenges: towards truly scalable ad hoc networks," in *proc. MobiCom*, 2007, pp. 207–214.

[17] Ram Ramanathan, "Challenges: a radically new architecture for next generation mobile ad hoc networks," in *Proc. MobiCom*, 2005, pp. 132–139.

[18] Lichun Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *Proc. MobiCom '01*, New York, NY, USA, 2001, pp. 210–221, ACM Press.

[19] L. Bao and J.J. Garcia-Luna-Aceves, "Hybrid channel access scheduling in ad hoc networks," in *Proc. of IEEE ICNP 2002*, 2002.

[20] Z. Tang and J.J. Garcia-Luna-Aceves, "A protocol for topology dependent transmission scheduling," in *Proc. IEEE WCNC*, Sep. 1999.

[21] Carlos H. Rentel, *Network Time Synchronization and Code-based Scheduling for Wireles Ad Hoc Networks*, Ph.D. thesis, Carleton University, 2006.

[22] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," IETF Internet draft, draft-ietf-manet-olsr-06.txt, Sept. 2001.

[23] Scalable Network Technologies, "The qualnet simulator 4.0," .

[24] Bruce Mah, "An internet simulated atm networking environment," http://www.employees.org/~bmah/Software/Insane/.