AN ALGEBRAIC MODEL OF AN ADAPTIVE EXTENSION OF DIFFSERV FOR MANETs

Osman Salem and Abdelmalek Benzekri
Institut de recherche en informatique de Toulouse,
Université Paul Sabatier,
118 Route de Narbonne - 31062 Toulouse Cedex 04 – France
E-mail: {benzekri, osman}@irit.fr

Abstract: In this paper, we propose an extension to DiffServ QoS architecture in order to enhance its performance and its flexibility when used in MANETs and its adaptation to the characteristics of these networks. Then we present a formal model of our proposed extension using stochastic process algebras in order to verify the correctness and the efficiency of the proposed extension.

Key words: QoS; MANET; DiffServ; Admission Control; Stochastic Process Algebras; Markov chain; Performance Evaluation.

1. INTRODUCTION

A Mobile Ad Hoc Networks (MANETs) [1] is an autonomous system of mobile hosts connected by wireless links and forming a temporary network without any pre-existing infrastructure. Each host is directly connected to hosts that are within its range of transmission and reception, and it is free to move randomly in and out of any other host's range. Communication between hosts that are not located in the same covering range can be realized by establishing a multi-hop route through intermediates hosts that act as routers when they forward data for others.

A lot of research has been done in routing area, and today routing protocols are mature enough to face frequently changing network topology. A quick look at intended applications area for MANET shows the need to integrate real time

multimedia traffic with data traffic. Many QoS aware routing protocols and models that claim to provide a partial (or complete) solution to QoS routing problems have appeared, e.g. QoS-AODV [2], MP-DSR [3], ASAP [4], CEDAR [5].

In the current days, the Integrated services (IntServ) [6] and the differentiated services (DiffServ) [7] are the two principal architectures proposed to provide QoS in wired network. IntServ suffers from well-known scalability problem caused by massive storage cost at routers when keeping flows' state information. The migration of this architecture to MANETs is judged very heavy because of network's constraint in term of storage capacity, contention of RSVP's out-band signaling packets with data packets and the two ways reservation mechanism of RSVP. The two-way reservation mechanism is inadequate and slow to adapt with the highly dynamic nature of hosts in MANETs, which leads to frequently change in the paths and thus rendering existing reserved resources unusable for some amount of time, in addition to excessive control overhead when path is broken.

DiffServ [7] on the other hand classify flows into several classes whose packets are treated differently in forwarding routers. It was designed to overcome the scalability drawbacks of IntServ. However, the notion of three kinds of nodes (ingress, interior, and egress nodes) and the SLA [7] (service level agreement) do not exist in MANETs. In DiffServ, the edge router is responsible to mark DSCP for each flow according to user profile listed in the SLA that includes the whole or partial traffic conditioning rules used to mark or re-mark traffic streams, discard or shape packets according to the traffic characteristics such as rate, and burst size. To alleviate these problems in MANETs, each host must be able to act as an edge and core router, and each host must be responsible for marking its traffic with the appropriate DSCP according to application's requirements. This means that every host plays the role of ingress router if it is transmitting data, a core router if it is forwarding data and an egress router if it is receiving data.

Several QoS schemes that are either a modification of the conventional IntServ and DiffServ based models have proposed for MANETs, like INSIGNA [8], FQMM [9], and SWAN [10]. SWAN (Service differentiation in stateless Wireless Ad Hoc Networks) differentiates traffic into 2 classes: high priority for real time UDP traffic and low priority for best effort UDP/TCP traffic. SWAN architecture (presented in figure 1) uses traffic differentiation in conjunction with a source based admission control mechanism to provide soft QoS assurances for real time traffic. However, this model differentiates traffic into two classes only; as it serves all real time traffics with equal priority, also it drops real time traffic with equal probability when congestion occurs, regardless their requirement in term of bandwidth and delay.

Many priority levels are required to differentiate important flows from others like in DiffServ, because it supports many classes of traffic. In addition, DiffServ relays on TCP rate control to reduce congestion and it is interesting enough to deserve a closer investigation. We think that the adoption of DiffServ by MANETs is better than IntServ due to characteristics of MANETs, which can not guarantee

any tight bounds on performance measures. It is useless to make a reservation of resources to guarantee a worst case delay for high priority flows in MANETs, because we can not guarantee neither the lifetime of the link or the delay on the link. Consequently, IntServ and reservation based approaches are not a favorite candidate to provide QoS in ad hoc networks. In contrast, DiffServ overcomes these disadvantages; it does not define any absolute guarantee and only proposes differentiations in scheduling when forwarding flows. In addition, extending DiffServ to Ad Hoc networks will provide consistent end-to-end QoS behavior when relaying flows between heterogeneous networks.

However, the differentiated services architecture does not define any scheme for taking corrective action when congestion occurs, and this is why a pure static DiffServ model is not suitable for ad hoc networks. Therefore, it is imperative to use some kind of feedback as a measure of the conditions of the network to dynamically regulate the traffic of the network when using this technology.

Our approach to provide QoS in MANETs is to extend DiffServ by adopting some positive aspects of SWAN and by adding new component to make DiffServ flexible and adaptive with bandwidth variation.

The qualitative and quantitative study of our scheme is conducted on a formal description, expressed through stochastic extensions of process algebras that allow us to formally describe both functional and performance aspects.

The rest of this paper is organized as follows. Section 2 gives a brief overview of the SWAN architecture. Our proposed scheme is described in detail in section 3. In section 4 we present a formal specification of our proposed extension to DiffServ using stochastic process algebra and we analyze both qualitative and quantitative aspects of the described model. Finally, in section 5 we conclude the paper with a summary of the results and future direction.

2. SWAN MODEL

The SWAN model presented in [10] differentiates flows into real time and best effort. This model works as follows: if a real time application at a node wants to communicate with another, it probes the network to obtain the minimal available bandwidth on the path, assuming the QoS aware routing protocol has found a path.

SWAN is composed from a classifier and a shaper located between the IP and the MAC layer as appear in figure 1. Aware that the bandwidth probe is sent at the beginning, so topology changing due to mobility and the variation of channel load conditions, SWAN uses rate control and explicit congestion notification mechanisms to adapt to these variations. It uses rate control for shaping or delaying the UDP/TCP best effort traffics and marks the ECN bits in the packet header to dynamically regulate admitted real-time traffic when temporary overload occurs by asking one or more real time source(s) to find another path. The rate control

operation of best effort traffic is performed at every mobile host in a distributed manner, where this rate will vary based on feedback information from MAC layer to maintain delay and bandwidth bounds for real time traffic.

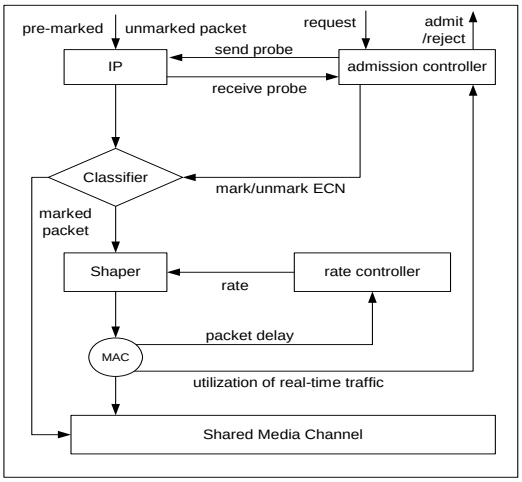

Figure 1. SWAN ARCHITECTURE

3. PROPOSED ARCHITECTURE

Our proposed extension to DiffServ QoS architecture is illustrated in figure 2. This scheme works as follows: applications notify their requirements and send their traffics to DiffServ component, which is responsible for marking and conditioning received packets from high level according to application's requirements. If received packets must be marked with one of the highest priority level, the marker component sends a request to the Call Admission Control (CAC) component. This request contains the amount of bandwidth required by this traffic to work properly. The CAC component verifies if the required QoS can be provided by the network, and this can be done by sending a probe request to routing protocol, which will collect the minimum end-to-end bandwidth (bottleneck bandwidth) along all existing paths from this source toward the specified destination. If the required bandwidth can not be provided by the network, CAC component reject the request and notify the application, which must decide to defer or to send with a medium priority profile.

In MANETs, even though admission control is performed to guarantee enough available bandwidth before accepting high priority flows, the network can still experience congestion due to mobility or connectivity changes. We do not attack the problem of broken link due to mobility, and we rely on the underlying routing protocol to detect and to resolve this problem by finding an alternate path toward the destination. Nevertheless, we try to resolve the problem of invading the channel by

others nodes forwarding medium and low priority flows and causing degradation to the QoS of accepted highest priority flows because of the limited bandwidth of this shared media. This is why the congestion control component is extremely important in our model. It monitors the network bandwidth utilization continuously and signal network congestion to the rectification component.

The proposed scheme has six basic components, namely bandwidth estimator, routing protocol, call admission control, DiffServ, congestion control and rectification component.

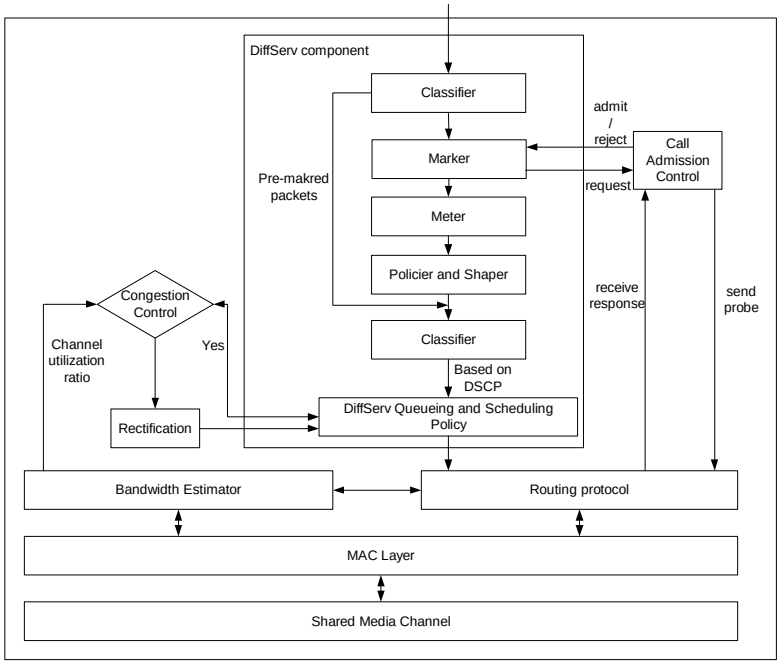

Figure 2. Proposed scheme

3.1 Bandwidth estimator component

The bandwidth estimator will periodically calculate the available bandwidth at each node. This value will be used by QoS aware routing protocol and the call admission control component to determine if flows can be admitted with one of the highest priority classes.

In MANETs, the communication media is shared among neighboring nodes, so to determine available bandwidth capacity in each node, we must take into account the transmissions of all its neighbors. We use the status of the shared channel as our base to calculate the channel utilization rate and the available bandwidth by each node. Using a shared channel allows mobile hosts to listen to packets sent within its

radio transmission range and to calculate resource availability. The channel utilization ratio is defined as the fraction of time within which a node is sensing the channel as being utilized. A node can calculate the utilization ratio of the channel by adding the time when it pumps data in the channel with the time that it finds the channel busy during period T, as $R = \frac{channel - busy - period}{T}$. Then we use the RTT method to estimate the average utilization ratio through the following formula: $Average\_Ratio_t = \alpha \times Average\_Ratio_{t-1} + (1-\alpha) \times measured\_Ratio_t$, where α is a constant belong to [0,1]. After estimating the channel utilization at time t, we are able to calculate the available bandwidth of a node at time t as $BW_t = W \times (1-R_t)$, where W is the raw channel bandwidth (2 Mbps for a standard IEEE 802.11 radio).

3.2 Call admission control component

The bandwidth information is already available at each node like explained in previous section. When the call admission control component receives a request to open a new session, it notifies the routing protocol component that is responsible for collecting the end-to-end bandwidth available along existing paths. When receiving route reply (RREP) packet for a path with available bandwidth greater than or equal to the requested value, then the session is admitted.

The objective of this CAC algorithm is twofold: to grant highest bandwidth utilization and avoiding at the same time the occurrence of congestion events by rejecting some requests. Nevertheless, it cannot guarantee that the bandwidth will not degrade for the admitted flow, since the available bandwidth may change after flows are admitted, due to the transmission of medium or low priority flows that do not need CAC by nodes sharing the media with any node belong to the path. Therefore, the network congestion can still occur and can be detected by the congestion control component.

3.3 Congestion control component

The role of the congestion control component is to detect network congestion, which is very simple to be detected in wired networks by verifying if the queue is overflow or even begin to build up. However, detecting congestion in MANETs is very difficult, because the queue length is no longer a valid indication of congestion. The MAC layer usually retries to transmit a packet for a limited number of times (e.g. default retry time of the IEEE 802.11b DCF is 7) before dropping this packet. Therefore, the queue may not have yet build up at the early stage of congestion. In our scheme, the congestion control component uses the channel utilization ratio provided by the bandwidth estimator component to detect congestion in MANETs. This component contains a predefined threshold value for channel utilization ratio, and compares this value with that provided by the estimator. If the estimated value is larger than this threshold, it assumes that due to congestion and signals congestion

occurrence to the rectification component, only if this node contains a high priority packet to forward.

Some neighbors of this congested node may be carrying medium or low priority traffic that reduces the available bandwidth and cause severe performance degradation to the high priority flows forwarded by this node. As they are in the direct reception range, these nodes can be directly informed by the rectification component.

3.4 Rectification component

The rectification component reacts to the detection of bandwidth degradation when there are some accepted flows with high priority crossing this congested node. Generally, when channel utilization ratio exceeds the predefined value, rectification component receives a notification message from the congestion control component and it broadcasts a stop-sending-request with TTL equal to 1. All nodes within direct reception range will receive this message and react accordingly if they forward medium or low priority packet.

Obviously, if all nodes receiving this request stop sending, there will be an under utilization of the channel and this strategy will prohibit these flows from using existing path even if there is enough bandwidth. To fight this problem, when a node receives stop-sending-request message, it does not immediately stop sending medium and low priority packets. Rather, each node chooses an exponential random amount of time from a pre-defined interval and triggers a counter with this value and begins to count down. If the counter reaches zero then this node drops all medium and low priority packets that are in the queue of this node and notify its neighbors that it stops sending. Also, this node (if it is not the source of these flows) mimics a broken link for these traffic by sending route error packet (RERR) to their corresponding sources, and it stops forwarding any new RREQ for a predefined amount of time to make this flows re-routed and dispersed away of the congested area. When neighbor nodes in the same range receive this notification (which can be embed in RERR), they cancel their counter by stops counting down.

3.5 DiffServ component

DiffServ divides traffics into many classes by marking a field in the IP packet header, called the Differentiated Services Code Point (DSCP) field. Its value depends on the traffic profile indicated by the application. The sender of a flow marks all outgoing packet in DS field of their IP headers and intermediate nodes forward these classes with different priorities with respect to the DSCP field content.

DiffServ supports 3 types of services: premium service, assured forwarding and best effort. Premium service or expedited forwarding (EF) class is used for loss and delay sensitive applications such as voice over IP (VoIP). Assured forwarding

classes offer a lower priority service from the previous one (EF), and each class of this four is subdivided into three subclasses (3 priority level) [14]. Finally best effort traffic does not require any QoS guarantee. The DiffServ model is composed from a classifier and a traffic conditioner like appears in left part of figure 3. The traffic conditioner presented in right part of figure 3 is composed from a meter, shaper/dropper, maker, separate physical queues for each class of traffic and a scheduler to schedule packets out of queues. DiffServ enqueues flows in separate buffer where packets with high priority will be serviced out of the buffer before than packets marked with low and medium priority. In addition, low priority packets may be selectively dropped prior to dropping packets of medium and high priority when congestion occurs.

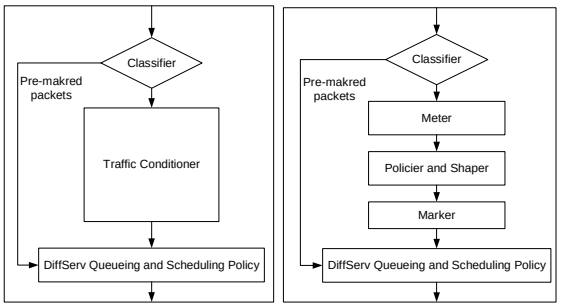

Figure 3. DiffServ router.

3.6 Routing protocol component

The first essential task of routing protocol is to find a suitable path through the network between the source and destination, and to record the minimum available bandwidth for CAC component. Our scheme does not rely on any specific reactive routing protocol and it may use any one able to collect the bottleneck value of the bandwidth along existing path.

4. FORMAL SPECIFICATION

One approach to ensure the correctness of our scheme is to use a formal model that integrates the theory needed for the verification of qualitative requirements together with the expressivity needed to analyze performance aspects. Moreover, a formal integration of these aspects in the initial phase of the system design allows the study of the potential dependencies that may occur between them.

Stochastic process algebras (SPA) [11] are formal specification languages used to describe the behavior of a system in order to derive its functional and

performance properties. They are suitable for automatic analysis and verification of the behaviour of systems. Several SPA languages have appeared in the literature; these include PEPA (Performance Evaluation Process Algebra [12]), TIPP (TImed Processes and Performability evaluation [13]) and EMPA (Extended Markovian Process Algebra [14]). They are a high level modeling formalism able to derive the state space structure that consists of all states that a system model can reach. Given the state space of the system, there are many practical algorithms for answering some verification and analysis questions.

These three languages propose the same approach to quantitative analysis, where a random variable is associated with each action to represent its duration. In this paper, we will use EMPA language, which is supported by a tool called TwoTowers. EMPA is inspired from PEPA and TIPP languages and it is considered like an extension to existing languages. The syntax of EMPA can be summarized by the following expression:

$$P = 0 \mid <a, \lambda>.P \mid <a, \infty_{L,W}>.P \mid <a, *>.P \mid P/L \mid P[\varphi] \mid P + Q \mid P \parallel_S Q \mid A$$

In the rest of this paper, we suppose that reader is familiar with process algebra. Interested reader must refer to [14] for further details.

We exploit the compositionality and abstraction features to specify each component apart by describing tasks that must be accomplished by this component. In addition, for the sake of simplicity and to avoid well-known state space explosion problem, we model only three queues to hold high, medium and low priority flows. Routing protocol component like DSR is modeled in a previous work [15], and a detailed specification of DiffServ router can be found in [16].

The complete model is composed from seven components: call admission control (CAC), DiffServ, Queues, Scheduler, Bandwidth estimator (Bandwidth_EST), congestion control (CC) and rectification component. Its algebraic specification and the specification of each component are given below:

$$\text{Adaptive\_DiffServ} \triangleq (\text{DIFFSERV} \parallel_{COM} \text{CAC}) \parallel_{Arr} \text{QUEUES} \parallel_{Ser} \text{SCHEDULER} \parallel_{SR} (\text{Bandwidth\_EST} \parallel_{CUR} \text{CC} \parallel_{NC} \text{RECTIFICATION})$$

$COM = \{send\_request, send\_response\}$ $Arr = \{arrival_H, arrival_M, arrival_L\}$
$Serv = \{deliver_H, deliver_M, deliver_L\}$ $SSR = \{stop\_sending\_request\}$
$CUR = \{send\_CUR\_to\_cc\}$ $NC = \{notify\_rectification\}$

- DIFFSERV $\triangleq$ <packet_arrival, λ>.CLASSIFIER
  CLASSIFIER $\triangleq$ <check_header, φ>.<classification, ψ>.CLASSIFIER1
  CLASSIFIER1 $\triangleq$ <not_marked, $\infty_{1,P}$>.MARKER +
  <marked, $\infty_{1,1-P}$ >.SEND_TO_QUEUE
- MARKER $\triangleq$ <τ, $\infty_{1,P}$>.<mark_pkt, $\infty_{1,1}$>.TRAFIC_COND +
  <τ, $\infty_{1,1-P}$>.<send_request, $\infty_{1,1}$>.WAIT_DECISION
  WAIT_DECISION $\triangleq$ <send_response, *>.MARKER1
  MARKER1 $\triangleq$ <mark_pkt, $\infty_{1,p}$>.TRAFIC_COND +
  <notify_application, $\infty_{1,1-p}$>.MARKER

- TRAFIC_COND ≜ <shaping_and_conditioning, η>. SEND_TO_QUEUE
  SEND_TO_QUEUE ≜ <$arrival_H$, λ_H>.DIFFSERV + <$arrival_M$, λ_M>.DIFFSERV+
  <$arrival_L$, λ_L>.DIFFSERV
- CAC ≜ <send_request, *>.<prepare_probe, β>.
  <send_probe, $\infty_{1,1}$>.WAIT_RESP
  WAIT_RESP ≜ <receive_probe_response, δ>.ADM_MECHANISM
  ADM_MECHANISM ≜ <comparing_values, ε>.ADM_RESULT
  ADM_RESULT ≜ <accept, $\infty_{1,P}$>.<send_response, $\infty_{1,1}$>.CAC +
  <reject, $\infty_{1,1-P}$>.<send_response, $\infty_{1,1}$>.CAC
- QUEUES ≜ $Queue\_High_0 \|_\emptyset Queue\_Medium_0 \|_\emptyset Queue\_Low_0$
  $Queue\_High_0$ ≜ <$arrival_H$, *>.$Queue\_High_1$
  $Queue\_High_i$ ≜ <$arrival_H$, *>.$Queue\_High_{i+1}$ +
  <$deliver_H$,*>. $Queue\_High_{i-1}$ $1 \le i \le N-1$

  $Queue\_High_N$ ≜ <$deliver_H$,*>. $Queue\_High_{N-1}$
  $Queue\_Medium_0$ ≜ <$arrival_M$, *>.$Queue\_Medium_1$
  $Queue\_Medium_j$ ≜ <$arrival_M$, *>.$Queue\_Medium_{j+1}$ +
  <$deliver_M$,*>. $Queue\_Medium_{j-1}$ $1 \le j \le M-1$
  $Queue\_Medium_M$ ≜ <$deliver_M$,*>. $Queue\_Medium_{M-1}$
  $Queue\_Low_0$ ≜ <$arrival_L$, *>.$Queue\_Low_1$
  $Queue\_Low_k$ ≜ <$arrival_L$, *>.$Queue\_Low_{k+1}$ +
  <$deliver_L$,*>. $Queue\_Low_{k-1}$ $1 \le k \le P-1$
  $Queue\_Low_P$ ≜ <$deliver_L$,*>. $Queue\_Low_{P-1}$
- SCHEDULER ≜ <$deliver_H$, $\infty_{3,1}$>.<$service_H$, μ>.SCHEDULER +
  <$deliver_M$, $\infty_{2,1}$>.<$service_M$, μ>.SCHEDULER +
  <$deliver_L$, $\infty_{1,1}$>.<$service_L$, μ>.SCHEDULER +
  <stop_sending_request,*>.<$service_{SR}$, $\infty_{1,1}$>.SCHEDULER
- Bandwidth_EST ≜ <listen_to_channel, ω>.<cal_ch_util_ratio,θ>.
  <send_CUR_to_cc, $\infty_{1,1}$>.Bandwidth_EST
- CC ≜ < send_CUR_to_cc, *>.<compare_with_thrsh, $\infty_{1,1}$>.
  (<lower, $\infty_{1,P}$ >.CC + <higher, $\infty_{1,1-P}$ >.Congested_Channel)
  Congested_Channel ≜ <check_high_priority_queue, $\infty_{1,1}$>.(<empty, $\infty_{1,P}$ >.CC +
  <not_empty, $\infty_{1,1-P}$ >.<notify_rectification, $\infty_{1,1}$>.CC)
- RECTIFICATION ≜ <notify_rectification, *>.
  <stop_sending_request, $\infty_{4,1}$>.RECTIFICATION

4.1 Function analysis

The functional analysis aims at verifying the correctness of the designed model and at detecting conceptual error in its behaviour. An important advantage of using TwoTowers is the possibility of exploiting well knows formal verification techniques to investigate functional properties of the model. Indeed, with respect to

conventional simulation environments (e.g. Network Simulator NS-2), TwoTowers provides verification of the satisfaction of correctness properties. Prominent examples of functional verification are checking that specified model does not lead neither to deadlock nor to livelock, controlling that certain activities are carried out according to a given order, or ensuring that certain resources are used in a mutually exclusive way.

The technique adopted in this tool to provide functional verification support is based on temporal model checking logic like μ-calculus/computational tree logic (CTL) [17]. With model checking, we refer to the possibility to express functional requirements by means of a set of formulas and verifying the satisfaction of desired properties by the algebraic specification. We start our analysis by verifying some behavioural requirements that we have formalized through the following formulas:

- No_deadlock = (min X = [-]ff ∨ ⟨-⟩X)
- AG([$arrival_H$]ff ∨ [$arrival_H$]A(([$deliver_M$]ff ∧ [$deliver_L$]ff) W < $deliver_H$ > tt))
- AG([send_stop_sending_request]
  A([send_stop_sending_request]ff W < notify_rectification > tt)
  ∧ [notify_rectification]A([notify_rectification]ff W < higher > tt))

Informally a formula $\langle a \rangle \psi$ holds in a state if there is a-labeled transition from this state to another one where the property ψ holds, i.e. it expresses a possibility. The formula $[a]\psi$ holds in a state if the specified property ψ holds in each state that is reachable through a-labeled transition, i.e. it expresses a necessity.

The first equation verifies the freedom of our model from deadlock. Equation 2 ensures if there is a high priority packet in the queue, it will be served before low and medium priority packets. This is explained by the fact that for any state (operator global G) of any computation (operator A) starting at the initial state, action "$arrival_H$" can not be executed by a state or if it can be executed the derivative must verify this formula A(([$deliver_M$]ff ∧ [$deliver_L$]ff) W <$deliver_H$>tt) which ensures that high priority packet will be scheduled for transmitting before than medium and low priority packets. Finally, equation 3 ensures that stop_sending_request packet can not be transmitted by the rectification component before the notification of the congestion control component, and this notification can not be transmitted before the occurrence of congestion , which can be detected when the available bandwidth is great than a pre-defined threshold.

4.2 Performance analysis

QoS characterizes the non-functional properties of a system; it is expressed in terms of a number of quantifiable parameters that can be easily evaluated after the derivation of CTMC from algebraic specification of the model, via a structured operational semantics rules. This chain can be used for constructing the infinitesimal generator matrix which is used to calculate the steady-state and transient probability distribution for the system. These probabilities will be used to derive the desired

performance parameters by assigning a number describing a weight (usually called a reward) is attached to every state of the CTMC, and the performance parameter is defined as the weighted sum of the steady state probabilities of the Markovian model, like shows the following formula:

$$\text{Performance parameter} = \sum_{i=1}^{N} r_i \times \pi_i$$

Where N is the size of CTMC, π_i is the steady state probability of state i, and r_i is the reward attached to state i. EMPA provides an automatic way to calculate performance parameters using the preceding formulas, in addition to the implemented simulation routine. Using these techniques, we evaluate the performance of our algebraic model by focusing at the variation of average delay experienced by a packet and the throughput of the three modeled classes. We have increased the arrival rate of flows that pass through the model, and we divide this rate equally between all classes.

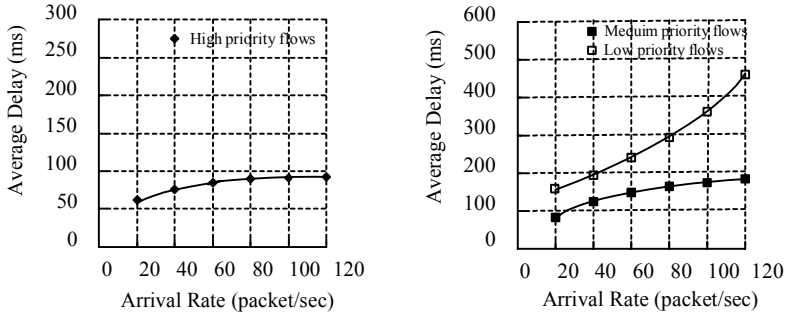

Figure 4. Average delay vs arrival rate

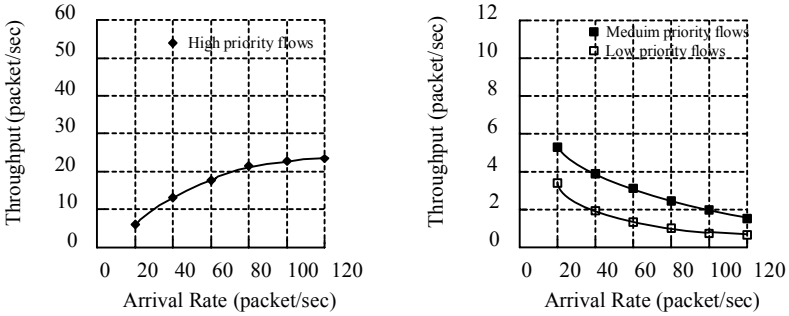

Figure 5. Throughput vs arrival rate

The variation curves that are presented in figures 4 and 5 show that our model can provide an acceptable delay and throughput for packets within highest priority range. The accuracy of these results is going to be depending on the detail that we have invested in the algebraic model, on the service rate of the scheduler, on the restriction of activities duration in SPA to exponential delay, and finally to the simple priority scheduler mechanism for queues instead of RED (Random Early Detection) usually used in DiffServ. These approximations are used to deal with the trade-off between accuracy of the model and the ability of the tool to verify functional behaviour due to limited memory with respect to the number of states in functional transition diagram (FTD), and to avoid state space explosion problem when evaluating performance parameters by using CTMC. The used scheduling mechanism is the responsible of the fact that medium and low priority traffics experience a long delay and their throughputs degrade exponentially to near zero.

5. CONCLUSION AND FURTHER RESEARCH

In this paper, we have described a new scheme to create an adaptive extension to DiffServ for MANETs after investigating the suitability of the existing QoS wired technologies (IntServ and DiffServ) to the characteristics of these networks. Our scheme is based on dynamic estimation of the available bandwidth by each node in its shared media in a completely distributed manner, and adopts the call admission control mechanism used in SWAN to gather minimum available bandwidth along existing paths towards a specified destination in route discovery phase. These additional components make DiffServ treats highest priority classes in an elegant and dynamic way while remaining stateless and lightweight.

Stochastic process algebras allow only the description of the task that must be accomplished by each component with an exponentially distributed duration in order to derive the CTMC from the described model. However, this is seen unrealistic for components that need deterministic duration and a description of this scheme with generally distributed activities will provide more accuracy while evaluating its quantitative parameters. This is why we focus in our current and future work at specifying our proposed scheme with generally distributed activities. In addition, some tasks need to be refined like the mechanism executed by nodes when receiving stop-sending-request, because the naïve suppression of any routing information by the node that decides react face to network overload is not a good strategy and may prevent the use of the only possible path between two hosts. This why there is a needs to enhance this mechanism in our future work.

REFERENCES

[1] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, January 1999.

[2] Charles E. Perkins and Elizabeth M. Belding-Royer, "Quality of Service for Ad Hoc On-Demand Distance Vector", Internet Draft, draft-perkins-manet-aodvqos-02.txt, 14 October 2003.

[3] Roy Leung, Jilei Liu, Edmond Poon, Ah-lot Charles Chan, Boachun Li, "MP-DSR: A QoS Aware Multi Path Dynamic Source Routing Protocol For Wireless Ad Hoc Networks", in the Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN'01), November 14 - 16, 2001, Tampa, Florida.

[4] Jianbo Xue, Patrick Stuedi and Gustavo Alonso, "ASAP: An Adaptive QoS Protocol for Mobile Ad Hoc Networks", in the Proceeding of 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC 2003), 7-10 September 2003, Beijing, china.

[5] R. Sivakumar, P.S Inha and V. Bharghavan, "CEDAR: A Core-Extraction Distributed Ad Hoc Routing algorithm", IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp. 1454-1465, August 1999.

[6] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the internet architecture: an overview", RFC 1633, USC/Information Sciences Institute, MIT, Xerox PARC, June 1994.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[8] S-B. Lee and A.T. Campbell, "INSIGNIA: In-band Signaling Support for QoS in Mobile Ad Hoc Networks", in the Proceedings of 5th International Workshop on Mobile Multimedia Communications (MoMuC, 98), Berlin, Germany, October 1998.

[9] H. Xiao, W.K.G. Seah, A. Lo and K.C. Chua, "A Flexible Quality of Service Model for Mobile Ad Hoc Networks", in the Proceedings of IEEE Vehicular Technology Conference (IEEE VTC2000-spring), 15-18 May 2000, Tokyo, Japan, pp 445-449.

[10] Gahng–Seop Ahn, A. T. Campbell, A. Veres, and Li-Hsiang Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks", in the Proceedings of IEEE INFOCOM 2002, June 2002.

[11] Ed Brinksma and Holger Hermanns 2001, "Process Algebra and Markov Chains", Lecture on Formal Methods and Performance Analysis, Nijmegen, pp183–231.

[12]Jane Hillston, "PEPA Performance Evaluation Process Algebra", Technical Report of Computer Science, Edinburgh University, March 1993.

[13] Herzog U. 1993, "TIPP: A Language for Timed Processes and Performance Evaluation", Proceedings of the First International Workshop on Process Algebra and Performance Modelling, University of Edinburgh, UK.

[14] Bernardo M. and R. Gorrieri 1998b, "A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time", Theoretical Computer Science, pp1-54.

[15] Abdelmalek Benzekri and Osman Salem, "Modelling and Analyzing Dynamic Source Routing Protocol with General Distributions", In the Proceedings of the 11th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA04), 13-16 June 2004.

[16] Abdelmalek Benzekri and Osman Salem, "Functional Modelling and Performance Evaluation for Two Class DiffServ Router using Stochastic Process Algebra", in the Proceeding of the 17th European Simulation Multiconference, Nottingham - UK, SCS-European Publishing House , PP. 257-262, 10 June 2003.

[17] E.M. Clarke, E.A. Emerson and A.P. Sistla, "Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications", in ACM transaction on programming languages and systems 8, pp. 244-263, 1986.