Cluster Computing 7, 91–103, 2004

Resource Management for Ad-Hoc Wireless Networks with Cluster Organization

IONUT CARDEI *, SRIVATSAN VARADARAJAN, ALLALAGHATTA PAVAN and LEE GRABA
Honeywell Labs, 3660 Technology Dr., Minneapolis, MN 55418, USA

MIHAELA CARDEI and MANKI MIN
University of Minnesota, 200 Union St. SE, Minneapolis, MN 55455, USA

Abstract. Boosted by technology advancements, government and commercial interest, ad-hoc wireless networks are emerging as a serious platform for distributed mission-critical applications. Guaranteeing QoS in this environment is a hard problem because several applications may share the same resources in the network, and mobile ad-hoc wireless networks (MANETs) typically exhibit high variability in network topology and communication quality. In this paper we introduce DYNAMIQUE, a resource management infrastructure for MANETs. We present a resource model for multi-application admission control that optimizes the application admission utility, defined as a combination of the QoS satisfaction ratio. A method based on external adaptation (shrinking QoS for existing applications and later QoS expansion) is introduced as a way to reduce computation complexity by reducing the search space. We designed an application admission protocol that uses a greedy heuristic to improve application utility. For this, the admission control considers network topology information from the routing layer. Specifically, the admission protocol takes benefit from a cluster network organization, as defined by ad-hoc routing protocols such as CBRP and LANMAR. Information on cluster membership and cluster head elections allows the admission protocol to minimize control signaling and to improve application quality by localizing task mapping.

Keywords: adaptive resource management, QoS adaptation, ad-hoc wireless networks, cluster-based routing

1. Introduction

Recent years have seen significant advances in the area of wireless networks, enabled by component miniaturization and improvements in processing power. Ad-hoc wireless networks have opened new opportunities for applications needing rapid deployment and mobility, such as military and civilian search and rescue missions. The US government, through the Office of Naval Research and DARPA, has initiated significant research into integrating ad-hoc wireless networks into the modern battlefield. The envisioned scenarios include a mix of mobile communication platforms, UAVs, UGVs, ships, ground vehicles, airplanes and ground units executing a range of collaborative Command, Control, Intelligence, Surveillance and Reconnaissance (C2ISR) missions. The mission-critical distributed applications run in this environment have strict demands on Quality of Service (QoS) while communication and processing resources (CPUs) may be shared concurrently. Over time, application importance as well as resource demands may vary due to mission mode changes. Thus, conflicting demands for resources must be arbitrated online in order to preserve the QoS for critical real time applications. In most cases application admission and resource arbitration are implemented by a resource management system.

The resource manager performs resource allocation and directs applications to adapt to resource availability, which is key to supporting a dynamic environment. The nature of communication in ad-hoc wireless networks complicates implementation of resource management. Hop-by-hop message forwarding, node mobility and signal quality add great variability to link quality, in terms of availability, bandwidth and delay. The variation in resource availability is challenging for guaranteeing QoS constraints.

In this paper we introduce DYNAMIQUE, a middleware for QoS specification, end-to-end negotiation and adaptive resource management, designed for ad-hoc wireless networked environments. At the foundation of DYNAMIQUE are the notions of QoS negotiation, translation and adaptation derived from our earlier work on RTARM [2].

The applications supported by our work have a pipeline topology with periodic communicating processing stages. The resource model we designed is more general and is appropriate for applications with arbitrary graph topologies. This paper specifically addresses the problems of application admission control, resource allocation and adaptation. During admission control, applications negotiate QoS contracts with the resource management system. A QoS request includes a range of Quality of Service supported by the application. The resource manager (RM) translates a QoS request into a request for system resources, such as network bandwidth and CPU cycles, and allocates resources for the new application considering availability and resources allocated to other applications. DYNAMIQUE uses a multidimensional representation for QoS, with one dimension for each resource type, e.g., network bandwidth, memory or CPU load. An admission

* Corresponding author.
E-mail: Ionut.Cardei@honeywell.com

request indicates a range of QoS that is feasible for the application, between minimum acceptable and a maximum usable. Intervals permit the allocation algorithm to be more flexible by allowing applications to negotiate a lower QoS when resources are not available to satisfy the maximum desired QoS.

Resource allocation has the goal of maximizing application Quality of Service and resource utilization. The level of QoS satisfaction for applications is dependent on resource availability and application criticality (importance). Information about allocated resources per application is stored into a QoS contract that changes only after the system performs external adaptation, which is one of two types of application adaptation. It involves lowering the resource allocation for existing applications so that a new application with higher criticality can be admitted in the system (QoS contraction). Also, applications are externally adapted when they are allocated extra resources made available by other applications departing the system (QoS expansion). The second type of adaptation is called Feedback Adaptation [1] and is similar to closed loop control. The RM continuously monitors the application and uses the difference between the measured QoS and the contracted QoS to adjust the application operation parameters without changing the resource allocation or the negotiated QoS contract.

In this paper we considered the particular nature of ad-hoc wireless networks and we designed an architecture that is suitable for environments that do not benefit from the features present in wired networks, such as abundant bandwidth, low latency and connection stability. Motivated by node mobility and communication quality variability, we designed a distributed resource management architecture. The admission protocols attempt to minimize the state information that is stored in the mobile nodes, as well as to minimize the control protocol overhead. In addition, the admission protocols take advantage of network topology state information available from the routing protocol that is executed on all nodes. In particular, the admission protocol is built on the notion of *clusters*, collections of nodes that define a local routing domain in ad-hoc wireless routing protocols such as LANMAR [4] and CBRP [8]. Routing in large mobile ad-hoc wireless networks (MANETs) is more scalable when nodes have a hierarchical organization. Locality and mobility patterns or logical association define cluster formation and membership and we assumed similar relationships for components of distributed applications. Thus, we expect to achieve better application QoS by accounting for this factor. The admission protocol and the resource allocation algorithm attempt to admit application tasks first in the same cluster, in order to minimize presumably costly inter-cluster communication. Within a cluster, the greedy admission protocol maps stages on the same node, with the intention to minimize inter-stage communication altogether. Cluster management (membership selection and cluster head election) is solely the responsibility of the routing protocol and is not addressed in this paper.

At this phase in our project we have defined resource models and designed protocols for admission control. We are currently evaluating the protocol performance through simulation. Initially we considered the problem of admission and resource allocation for multi-stage pipeline applications in an ad-hoc wireless network, more specifically, the optimal mapping of stages on network nodes, depending on resource demands (CPU and network bandwidth) and availability.

We present a resource model for application admission with a multidimensional Quality of Service representation and we formalize the resource allocation problem for single application and multi-application admission as a mixed linear-integer program optimization in section 2. We then present a greedy approach and a distributed protocol for admission with adaptation in section 3. Section 4 describes simulation results for the admission protocol and approximations for optimal allocation. Section 5 concludes the paper with some final remarks.

1.1. Related work

Resource management for wired networked environments is a mature area of research. The DARPA Quorum program has sparked multiple projects that are focused on frameworks and protocols for QoS provisioning for real-time distributed systems. The Amaranth project [5,10], from CMU, builds a user-centric architecture for multi-dimensional, adaptive, assured Quality of Service for heterogeneous distributed systems. The system accepts admission requests for multiple resources within a range of *discrete* ordered values (e.g., standard audio sampling rates, image color depth or 56, 64 and 128 bit encryption levels), and attempts to maximize system utility with respect to application utility considering multiple performance levels for each QoS dimension. Lee proposes a polynomial-time approximation based on convex hull frontiers, with near optimal utility for the NP-hard resource allocation problem for multiple resources and QoS dimensions.

A precursor of DYNAMIQUE is the Real-time Adaptive Resource Management framework (RTARM) [2]. RTARM defines a hierarchical architecture of Service Managers that supports end-to-end QoS negotiation, translation and adaptation for distributed applications in wired networks. Service Managers at the bottom of the hierarchy are responsible for managing basic resources (CPU, memory or network resources). Higher-level Service Managers aggregate services from lower-level managers and provide services for end-to-end QoS provisioning. RTARM introduces a multidimensional representation for QoS and supports plug-in for third-party resource managers.

The Global Resource Management System project introduces the ripple scheduling admission and adaptation protocol [7] and a multi-resource QoS model that inspired the DYNAMIQUE approach [6].

Mobile ad-hoc wireless networks are characterized by link volatility, dynamic topology and small factor, therefore resource management systems specifically designed for wired networks cannot be plugged in directly. New architectures and protocols are required to accommodate the wireless dynamic environment. A key component of any resource management system is the network RM. Guaranteeing delay and

bandwidth in MANETs is complicated by the network instability generated by node mobility, link quality variation and by resource scarcity. One of the notable architectures for network QoS for ad-hoc wireless networks is based on INSIGNIA [9], proposed by the COMET group at Columbia University. INSIGNIA uses in-band IP signaling to deliver adaptive real-time communication services. It provides protocols for flow setup, restoration and adaptation that allow applications to receive end-to-end QoS in situations when the end-to-end path experiences failures. INSIGNIA defines IP optional header fields to pass flow reservation state along the route and to signal failures. The protocol overhead is minimized by piggybacking control information within the data packets.

2. Resource allocation model

The DYNAMIQUE resource management system supports periodic distributed applications with a pipeline topology consisting of a chain of communication tasks, although the resource model presented in this section applies more generally to distributed applications with arbitrary graph topologies. An example of a pipeline application is illustrated in figure 1.

In this figure, the video camera on board the Predator UAV captures periodic frames with potential targets. The images are manipulated by two video processing and decision stages on the intermediate nodes. The final fire command for a recognized target is issued to the howitzer. This type of mission-critical applications demands strict limits on end-to-end latency and requires significant bandwidth for network connections. Our resource management protocols and algorithms allow resource sharing between multiple distributed applications with conflicting demands and different arrival times. During admission control, the user submits a QoS request to the system, consisting of a range of feasible QoS, between minimum acceptable and a maximum value. The RM translates this QoS request into individual requests for system resources and computes an allocation for the new application, depending on current availability. The first allocation model we developed computes task mapping on the network nodes and resource allocation for one distributed application considering only current resource availability. Section 2.1 describes this resource model and section 4 resents simulation results. We formulate the resource allocation as an optimization problem with the objective to maximize the quality of the admitted application. The solution identifies the node where each task will be executed (task mapping) and how much resources have been allocated. The multi-application resource model, the second resource model we have developed and is currently under evaluation, determines a resource allocation and mapping for multiple applications.

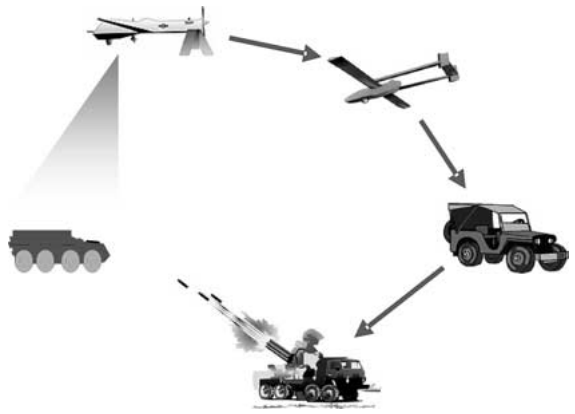

Figure 1. Distributed targeting application with pipeline topology.

We define two classes of resources. *Node resources*, such as CPU load, memory and energy, are related to a single node. A pair of nodes characterizes the other resource class, *network resources*. Bandwidth is a network resource and its specification requires a source–destination pair.

Both resource allocation models are based on these assumptions that simplify the problem formulation:

(1) QoS dimensions have a one to one correspondence to system resources types. A QoS contract for an application specifies for each resource needed a [min, max] range defining an acceptable range.

(2) Node resources are modeled as limited buckets of capacity $r^{\max}$ with the admission condition $\sum r_i \leqslant r^{\max}$, where r_i is the resource amount allocated for task i. The total resource utilization in the system cannot exceed the available amount.

(3) A network resource is modeled as a limited bucket associated to a pair of connected nodes (e.g., bandwidth).

(4) Network links are bi-directional. Connections in both directions share the same network resource.

(5) Resources are independent of each other.

(6) Resources are not probabilistic and the system guarantees the contracted QoS. We will consider management for resource with probabilistic availability and behavior in the later phases of the project.

The next section describes the single-application model.

2.1. Single application resource model

The model defines node and network resources, and formalizes allocation constraints. The allocation problem is formulated as an optimization problem.

Consider a distributed environment with n computing nodes and m node resource types and one network resource. A distributed application is composed by s communicating tasks, $T = \{T_1, T_2, \ldots, T_s\}$.

2.1.1. Node resources

The admission request for application T is described by a set of Quality of Service descriptors, one for each QoS dimension. The QoS request is described by matrices Q^m and Q^M. $Q^m = (q_{ij}^m)_{i=1,\ldots,m,\, j=1,\ldots,s}$ and $Q^M = (q_{ij}^M)_{i=1,\ldots,m,\, j=1,\ldots,s}$ define the minimum and, respectively, the maximum QoS requirements for the application tasks $1, \ldots, s$, where m is the

number of QoS dimensions (number of resource types) and $q_{ij}^m \leqslant q_{ij}^M$. In addition, q_1 is considered "a better QoS" than q_2 if $q_1 > q_2$.

Matrix R^0 describes the available resources before the application admission. $R^0 = (r_{ij}^0)_{i=1,\ldots,m,\, j=1,\ldots,n}$, where r_{ij}^0 is the available amount of resource of type i on node j; $R^0 \in [0, \infty)^{m\times n}$.

The admission control admits the s tasks in the system. The mapping of the s tasks on the n nodes is given by matrix

$$X = (x_{ij})_{i=1,\ldots,s,\, j=1,\ldots,n},$$

with

$$x_{ij} = \begin{cases} 1, & \text{if task } T_i \text{ was mapped (allocated) on node } j, \\ 0, & \text{otherwise;} \end{cases}$$

$X \in \{0, 1\}^{s\times n}$.

The vector Map is defined as $Map_i = j$ if task T_i was mapped on node j.

The resource management system allocates resources $R^a = (r_{ij}^a)_{i=1,\ldots,\, m,\, j=1,\ldots,s}$ to the s tasks. The amount $r_{ij}^a > 0$ of resource i has been assigned to task j.

2.1.2. The network resource

The network resource model assumes that network resources are allocated independently for each connection between any two nodes. The network resource is modeled by a limited bucket for each bi-directional pair of nodes (i, j). Matrix NR^0 defines the available network resource at admission time. $NR^0 \in [0, \infty)^{n\times n}$ and NR_{ij}^0 defines the network resource available to the (i, j) communication link.

Matrices NQ^m and NQ^M define the minimum, respectively, maximum network resource requirement for each connection (i, j), for which tasks T_i and T_j communicate; $NQ^M, NQ^m \in [0, \infty)^{s\times s}$.

The set TC contains all required connections between tasks; $TC = \{(i, j) \mid T_i \text{ communicates with } T_j,\ i < j\}$.

Matrix NQ^a defines the allocated network resource. NQ_{ij}^a = network resource allocated for connection (T_i, T_j); $NQ^a \in [0, \infty)^{s\times s}$.

This resource model assumes there is a (possible multihop) path in the network between any two nodes and that all resource allocations for connections are independent. The resource management system maps the tasks to nodes and allocates resources for each connection between two tasks. A connection between tasks T_i and T_j is mapped to a connection between nodes Map_i and Map_j. The system allocates NQ_{ij}^a resource to the (i, j) connection by subtracting the same resource amount NQ_{ij}^a from the available network resource for connection (Map_i, Map_j): after allocation,

$$NR^0[Map_i, Map_j]' = NR^0[Map_i, Map_j] - NQ_{ij}^a.$$

Matrix NR^a defines the allocated network resources for an application $(T_1, \ldots, T_s, TC)$:

$$NR^a = (nr_{xy}^a)_{x,y=1,\ldots,n},$$

where $nr_{xy}^a = nq_{ij}^a$ and $(x, y) = (Map_i, Map_j)$; $NR^a \in [0, \infty)^{n\times n}$.

Our model can accommodate alternate allocation strategies by adjusting the equations for resource constraints and resource availability update.

We are currently in the initial phase of evaluating another approach for modeling network resources, based on node communication capacity and explicit end-to-end path information that would be available from the routing protocol.

2.1.3. Allocation constraints

The next relations state the conditions that resource allocation must meet:

Node resource constraints

(1) *Task mapping*. Each task is admitted on exactly one node:

$$\sum_{j=1,\ldots,n} x_{ij} = 1, \quad \text{for all tasks } i = 1, \ldots, s.$$

(2) *Application QoS*. Allocated resources satisfy QoS requirements:

$$Q^m \leqslant R^a \leqslant Q^M.$$

(3) *Resource availability*. Resources allocation is limited by availability:

$$R^a X \leqslant R^0 \quad \text{or}$$
$$x_{1j} r_{i1}^a + x_{2j} r_{i2}^a + \cdots + x_{sj} r_{is}^a \leqslant r_{ij}^0,$$

for all resources $i = 1, \ldots, m$ and all nodes $j = 1, \ldots, n$.

Network resource constraints

(1) *Application QoS*. All connections between tasks must be allocated network resources between minimum required and maximum needed. Note that Map maps from NR^a to NQ^a; $NQ^m \leqslant NQ^a \leqslant NQ^M$.

(2) *Resource availability*. The allocated network resource for all connections cannot exceed the available limit; $NR^a \leqslant NR^0$.

2.1.4. Utility functions

The optimal allocation is designed to maximize the application QoS utility defined as a function of the combined QoS satisfaction ratio for all tasks. The application overall utility is a linear combination of the node task utility and the network utility.

The node task utility

The node utility of task T_i for resource j is normalized to

$$u_{ij} = \begin{cases} \dfrac{r_{ij}^a - q_{ij}^M}{q_{ij}^M - q_{ij}^m}, & \text{if } r_{ij}^M > r_{ij}^m, \\ 1, & \text{otherwise.} \end{cases}$$

Matrix $U = (u_{ij})_{i=1,\ldots,s,\, j=1,\ldots,m}$. The node utility of task T_i is a weighted sum of resource utilities:

$$u_i = w_1 u_{i1} + \cdots + w_m u_{im}, \quad \text{with } i = 1, \ldots, s.$$

Weights $w_j \geqslant 0,\ j = 1, \ldots, m$, and $\sum w_j = 1$. The application node utility vector is $V = (u_1 \ \ldots \ u_s)^T = UW$.

$$V \in \mathbf{R}_+^{*s\times 1}, \quad W = (w_1 \ w_2 \ \ldots \ w_m)^T \in \mathbf{R}_+^{*m\times 1},$$

$w_j \geqslant 0$ and $\sum w_j = 1$.

The network utility

The network utility *nu* for application (T, TC), is defined as:

- if $nq_{ij}^M = nq_{ij}^m$ or $i = j$ then $nu = 1$, awarding maximum utility for full allocation or for tasks mapped on the same node;
- if $nq_{ij}^M > nq_{ij}^m$,

$$nu = \sum_{i=1,\ldots,s} \sum_{j=1,\ldots,s,\ i\neq j} \frac{nq_{ij}^a - nq_{ij}^m}{nq_{ij}^M - nq_{ij}^m}.$$

Application utility

The utility for the multi-stage admitted application is defined as the weighted sum of node task utilities and the network utility:

$$v = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_s u_s + \alpha_{s+1} nu.$$

In matrix notation, $v = A(V \ nu)^T$, with $A = (\alpha_1 \ \ldots \ \alpha_s \ \alpha_{s+1})$, $\alpha_i \geqslant 0$, and $\sum_{i=1,\ldots,s+1} \alpha_i = 1$.

2.1.5. The optimal allocation problem for the single-application resource model

The problem is to determine the task mapping X, a node resource allocation NR^a and a network resource allocation NQ^a, so that:

(1) node resource and network resource constraints (section 2.1.3) are obeyed;

(2) the application utility, v (section 2.1.4) is maximized.

This is a mixed integer programming optimization problem with a fairly complex structure. The work in [3] shows that the resource allocation problem is NP-hard by reduction to the integer bin-packing problem. To overcome the complexity, we designed a greedy admission protocol suitable for online use that will be described in section 3.

2.1.6. Selection criteria for utility weights

Formulating the admission utility function using weighted sums allows us to inject user-defined policies and application semantics into the allocation process. For the node task utility function, the relation between the resource type weights w_j, may impact the contribution of the task to the overall application utility, thus being a factor to the final mapping of tasks to nodes. In essence, the weight w_j can be directly correlated to the relative importance the user assigns to a resource type j. The admission process will map tasks to nodes where allocation of specific resources contributes maximum utility. For instance, if the fraction $w_{\text{CPU}}/w_{\text{Memory}} = 1/3$, then the admission algorithm, in the process of optimizing the total application utility value, will be more likely to map tasks to nodes where the memory allocation is closer to the maximum required. Similarly, the weights α_i from the application utility formula can be adjusted by users to express preference towards maximizing utility of specific tasks or of the network bandwidth allocation (α_{s+1}).

2.2. Multi-application resource allocation model

This model extends the single application, multi-stage model presented above, to the multiple application case, where each application is characterized by a priority, common for all its stages. We also assume that an application is *admitted* provided all its stages are admitted to the system. A critical constraint that applies to this model is that higher priority applications are never preempted by lower priority applications.

The objective of the multi-application allocation is to maximize the QoS satisfaction level for higher-priority applications. Another possible optimization goal that may be considered is to maximize the overall number of admitted applications.

2.2.1. Application model

Assume the system has to admit and allocate resources for the set $A = \{A_1, \ldots, A_t\}$ of applications, *ordered increasingly on priority*. The system consists of n nodes with resource availability $R^0 = (r_{ij}^0)_{i=1,\ldots,m,\, j=1,\ldots,n}$, for m resources distributed on the n nodes, and NR^0 for network resources. Each application A_i is characterized by:

- priority $p_i > 0$. Assume $p_i \leqslant p_{i+1}$;
- node resource demands Q_i^m = minimum requested QoS and Q_i^M = maximum requested QoS. Note that QoS dimensions map 1-to-1 to resources;
- network QoS demands NQ_i^m and NQ_i^M.

The resource management system will map application tasks and allocate resources. The vector $Y = (y_1, \ldots, y_t)$ indicates the application admission:

$$y_i = \begin{cases} 1 & \text{if } A_i \text{ is admitted,} \\ 0, & \text{otherwise.} \end{cases}$$

And matrices X_i define individual task mapping for application i.

2.2.2. Global objective function

The global objective function for the multi-application model requires normalization for individual application utility values. We want the utility contributed by each application to the overall objective to be proportional to the QoS (amount of resources) received from the system, and not to the number of stages.

The application node utility for application i is therefore normalized on the number of stages, s_i, and the network utility is also normalized on the number of connections, $|TC_i|$:

$$v_i = \frac{\alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_{s_i} u_{s_i}}{s_i} + \frac{\alpha_{s_i+1} nu_i}{|TC_i|},$$

$$v_i = A\left(\frac{V}{s_i}; \frac{nu}{|TC_i|}\right)^T,$$

with weight factors $A = (\alpha_1 \ \ldots \ \alpha_{s_i} \ \alpha_{s_i+1})$, $\alpha_j \geqslant 0$, and $\sum_{j=1,\ldots,s_i+1} \alpha_j = 1$.

The global objective function assigns more weight to utility contributed by higher priority applications. The goal is to maximize v^*:

$$v^* = y_1 p_1 v_1 + y_2 p_2 v_2 + \cdots + y_t p_t v_t = \sum_{i=1,\ldots,t} y_i p_i v_i,$$

where $y_i \in \{0, 1\}$,

$$y_i = \begin{cases} 1, & \text{if application } i \text{ is admitted,} \\ 0, & \text{otherwise.} \end{cases}$$

2.2.3. Multi-application constraints

The set of constraints for the optimization problems relates to resource allocation and priority-based preemption. The resource constraints are an extension of those listed in section 2.1.3:

- The resources allocated to all applications should not exceed initial availability:

$$\sum_{i=1,\ldots,t} y_i R_i^a X_i \leqslant R^0,$$

for node resources, and

$$\sum_{i=1,\ldots,t} y_i NR_i^a \leqslant NR^0,$$

for network resources.

- The allocated resources should satisfy QoS demands for admitted applications:
For all $i = 1, \ldots, t$

$$y_i Q_i^m \leqslant y_i R_i^a \leqslant y_i Q_i^M,$$

for node resources, and

$$y_i NQ_i^m \leqslant y_i NQ_i^a \leqslant y_i NQ_i^M,$$

for network resources.

- The constraint stating that higher priority applications cannot be preempted by lower priority applications is formulated as:

$$y_i \leqslant y_{i+1}, \quad \text{for all } i = 1, \ldots, t-1.$$

(Note: we have assumed that $p_i \leqslant p_{i+1}$.)

2.2.4. Allocation problem for the multi application resource model

The allocation optimization problem can be summarized as follows:

maximize the global objective function:

$$v^* = y_1 p_1 v_1 + y_2 p_2 v_2 + \cdots + y_t p_t v_t = \sum_{i=1,\ldots,t} y_i p_i v_i,$$

where

$$v_i = \frac{\alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_{s_i} u_{s_i}}{s_i} + \frac{\alpha_{s_i+1} nu_i}{|TC_i|}, \quad i = 1, \ldots, t,$$

and for each application i:

- $u_i = w_1 u_{i1} + \cdots + w_m u_{im}$, with $i = 1, \ldots, s$, $w_j \geqslant 0$, $j = 1, \ldots, m$, $\sum w_j = 1$,
- and

$$u_{ij} = \begin{cases} \dfrac{r_{ij}^a - q_{ij}^M}{q_{ij}^M - q_{ij}^m}, & \text{if } r_{ij}^M > r_{ij}^m, \\ 1, & \text{otherwise,} \end{cases}$$

and

- if $nq_{ij}^M = nq_{ij}^m$ or $i = j$ (tasks on the same node) then $nu = 1$,
- if $nq_{ij}^M > nq_{ij}^m$,

$$nu = \sum_{i=1,\ldots,s} \sum_{j=1,\ldots,s,\ i \neq j} \frac{nq_{ij}^a - nq_{ij}^m}{nq_{ij}^M - nq_{ij}^m},$$

with the following constraints (t is the number of applications):

$$\sum_{i=1,\ldots,t} y_i R_i^a X_i \leqslant R^0, \quad \text{for node resources, and}$$

$$\sum_{i=1,\ldots,t} y_i NR_i^a \leqslant NR^0, \quad \text{for network resources.}$$

For all $i = 1, \ldots, t$:

$$y_i Q_i^m \leqslant y_i R_i^a \leqslant y_i Q_i^M,$$

for node resources, and

$$y_i NQ_i^m \leqslant y_i NQ_i^a \leqslant y_i NQ_i^M,$$

for network resources (note that *Map* maps from NR_i^a to NQ_i^a);

$$y_i \leqslant y_{i+1}, \quad \text{for all } i = 1, \ldots, t-1$$

(note: we assumed that $p_i \leqslant p_{i+1}$) and

$$y_i \in \{0, 1\}.$$

The resource allocation problem asks for computing matrices Y – admitted applications, X_i – task mapping for application i, R_i^a – allocated node resources and NR_i^a, network resource, for application $i = 1, \ldots, t$.

In a real world scenario, application admissions come at different times. Our greedy admission method considers QoS

adaptation for existing applications, practically adding the extra resources to the available resources pool. This can be modeled by the above multi-application admission method by forcing selected critical applications (assumed in execution before this admission) to be admitted, i.e., $y_i = 1$.

The large number of variables ($t + tsn + tsm + tnn$) for this mixed integer program, makes a real-time approximation with branch-and-bound or even with a linear program unfeasible. Section 3 introduces a greedy admission protocol for ad-hoc wireless networks with cluster-based organization. The next sections present another way to decrease the optimization problem complexity.

2.2.5. Admission with minimum QoS followed by expansion
An alternative goal for the admission procedure might be to maximize the overall number of accepted applications, higher priority application having precedence. Later, a QoS expansion phase would increase the QoS for admitted applications from remaining resources. We break the large optimization problem in two smaller pieces with fewer unknowns. The two phases of the admission process are described in sections 2.2.5.1. and 2.2.5.2.

2.2.5.1. Admission with minimal QoS
The first phase is admission with minimal QoS, so that the overall number of admitted applications is maximized.
The optimization problem can be formalized as:

maximize the objective function:

$$v^* = \sum_{i=1,\ldots,t} y_i$$

with the following constraints:

$$\sum_{i=1,\ldots,t} y_i Q_i^m X_i \leqslant R^0, \text{ for node resources, and}$$

$$\sum_{i=1,\ldots,t} y_i NR_i^m \leqslant NR^0, \text{ for network resources;}$$

for all $i = 1, \ldots, t$:

$y_i Q_i^m = y_i R_i^a$, for node resources, and

$y_i NQ_i^m = y_i NQ_i^a$, for network resources;

for all $i = 1, \ldots, t-1$:

$y_i \leqslant y_{i+1}$ and

$y_i \in \{0, 1\}$, $y_i = 1$ if application i is admitted.

By admitting applications at their minimum requested QoS we significantly decrease the solution space. Only the unknown matrices X_i and $Y = (y_1, \ldots, y_t)$, with elements in $\{0, 1\}$ have to be determined for this integer program.

2.2.5.2. QoS expansion
After phase I – admission with minimal QoS – we have determined indicators y_i and X_i. Phase II involves QoS expansion. The system allocates remaining resources to admitted applications, with preference to higher priority applications.

The QoS expansion can be formulated as a linear program in the following way:

maximize the objective function (QoS satisfaction ratio):

$$v^* = y_1 p_1 v_1 + y_2 p_2 v_2 + \cdots + y_t p_t v_t = \sum_{i=1,\ldots,t} y_i p_i v_i,$$

where

$$v_i = \frac{\alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_{s_i} u_{s_i}}{s_i} + \frac{\alpha_{s_i+1} nu_i}{|TC_i|},$$
$$i = k, \ldots, t$$

and for each application

- $u_i = w_1 u_{i1} + \cdots + w_m u_{im}$, with $i = 1, \ldots, s$, $w_j \geqslant 0$, $j = 1, \ldots, m$, $\sum w_j = 1$,
- and

$$u_{ij} = \begin{cases} \dfrac{r_{ij}^a - q_{ij}^M}{q_{ij}^M - q_{ij}^m}, & \text{if } r_{ij}^M > r_{ij}^m, \\ 1, & \text{otherwise,} \end{cases}$$

- and
  - if $nq_{ij}^M = nq_{ij}^m$ or $i = j$, then $nu = 1$,
  - if $nq_{ij}^M > nq_{ij}^m$,

$$nu = \sum_{i=1,\ldots,s} \sum_{j=1,\ldots,s,\ i \neq j} \frac{nq_{ij}^a - nq_{ij}^m}{nq_{ij}^M - nq_{ij}^m},$$

with the following constraints:

$$\sum_{i=k,\ldots,t} y_i R_i^a X_i \leqslant R^0, \text{ for node resources, and}$$

$$\sum_{i=k,\ldots,t} y_i NR_i^a \leqslant NR^0, \text{ for network resources;}$$

for all $i = k, \ldots, t$:

$Q_i^m \leqslant R_i^a \leqslant Q_i^M$, for node resources, and

$NQ_i^m \leqslant NQ_i^a \leqslant NQ_i^M$, for network resources.

The $ts(m+n)$ remaining unknowns for this linear program are matrices R_i^a, and NQ_i^a, $i = 1, \ldots, t$. We have not evaluate this approach, as for realistic application settings ($n > 10$ nodes, $s = 3$ stages, $m = 2$ (CPU and memory) and $t > 1$ application, a linear program is not yet feasibly approximated online on mobile, low-powered processors.

3. Admission architecture and protocol

This section presents the resource management architecture and the admission protocol for multi-application admission control.

3.1. System architecture

All nodes in the wireless network have the same resource management architecture, as illustrated in figure 2. The architecture consists of an Application Service Manager, Resource Managers, Applications and Clients.

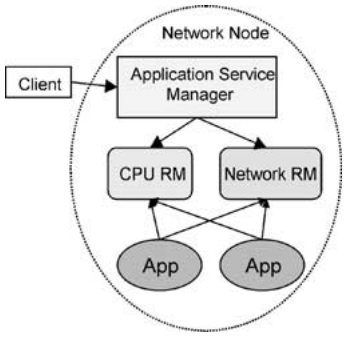

Figure 2. Resource management architecture at the node level.

Resource Managers manage a basic resource, such as CPU load, memory or energy. The Network RM controls the network resources, such as bandwidth and delay. All resource managers export a common interface for admission, adaptation and feedback adaptation that allows RMs for different policies to be simply plugged in.

The CPU resource manager administers the local CPU resource. Based on the current CPU resource allocation, it builds a process schedule and controls application CPU utilization. The CPU RM is a middleware layer wrapped on top of the local OS scheduler, or implements a real-time scheduling policy, such as RMA [11].

The Network RM controls bandwidth allocation, enforces traffic shaping and extracts network topology from the routing layer. As mentioned in the Introduction, the DYNAMIQUE admission protocol works in cooperation with a cluster-based ad-hoc routing protocol. Nodes are organized in clusters based on traffic locality, as in CBRP [8], or on logical membership and mobility patterns, as in LANMAR [4]. In any way, we assume the cost of communication within a cluster is lower than between clusters. The application admission protocol described in the next subsection attempts to improve admission utility by decreasing cost of communication, based on the assumption that communication in a MANET is less reliable while processing resources are plentiful.

The Application Service Manager (ASM) is responsible for the end-to-end resource management for a distributed application. It handles end-to-end QoS negotiation, admission and adaptation, by breaking end-to-end requests into individual contracts for basic resources passed to RMs (CPU and Network RM) and to other Application Service Managers in the network. ASMs receive admission requests from Clients that can be users, applications or other ASMs. Admission requests consist of a [min, max] range for acceptable QoS (CPU load, network bandwidth) for application tasks.

Applications in the DYNAMIQUE model consist of distributed periodic tasks connected in a pipeline topology. Depending on resource availability, tasks from the same application may be assigned on the same node, on different nodes in the same cluster, or on nodes from different clusters. Each case incurs an increasing cost for communication.

3.1.1. Location constraints

In a realistic environment some application tasks may have to be admitted on a specific node, or close to a geographical location. For example, an Automatic Target Recognition application requires the sensor and the target display stages to run on predetermined nodes, while the intermediary processing and recognition stages can be allocated on any node with sufficient resources. The current version of the DYNAMIQUE resource model does have specific handling for these situations. One can define "special" RMs for "radar" resources, and then the ASM will match the task request for a radar resource with the correct node. We think that the model can be easily adapted to support generic location constraints (for a specific node). Matrix X that defines the task mapping (see section 2.1.1) can be preloaded with x_{ij} values reflecting mapping of task i onto node j.

3.2. The multi-application admission protocol

We start this section with a user scenario that summarily describes an admission, focusing at the Application Service Manager level on the main line of events:

- *Actors*: a Client, Initiator ASM, CPU RMs, Network RMs.
- *Preconditions*: the MANET is established into clusters and connected; each node hosts: ASM, CPU RM, Network RM and application tasks.
- *Description*: the scenario starts when the Client sends an admission request to the ASM running on its node. This is called the Initiator ASM. The Initiator ASM forwards the request to the ASM on its local cluster head. The cluster head sends individual requests for resources to ASMs running in the local cluster. The ASMs (including the Initiator ASM) reply to the Initiator ASM with resource availability from local CPU and Network RM. If not enough resources to admit all tasks, the Initiator will forward requests to ASMs on selected cluster heads. These will forward the requests further to member nodes' ASMs that will reply to the Initiator with resource availability. When the Initiator has acquired enough resources to admit all tasks, it sends a commit message to all nodes that will host tasks. Then the Initiator replies to the Client with the admission result. The Client starts the Application and the system allocates resources.

Application Service Managers on cluster head nodes manage a "preference list" of clusters that defines the order in which admission requests are forwarded to remote clusters. The current version of the admission protocol implements a simple round-robin policy. More complex versions may coordinate for load balancing or optimize for resource availability. This is an entire area for optimization that we have not explored yet.

3.2.1. External adaptation

The admission protocol implements external adaptation. When not enough resources are available to admit all tasks, the ASM selects lower criticality applications and adapts them, by shrinking their QoS. The freed resources are added to the available resources and admission is attempted again. If admission is now feasible, the tasks are allocated resources

and the adapted applications lower their resource utilization. At this stage, the QoS expansion protocol is work in progress and will be presented at a later time.

3.2.2. Design goals

The admission protocol has three design goals:

3.2.2.1. Provide a good quality solution to the admission problem

An optimal solution for the resource allocation problem (section 2) is not feasible to compute online for the use in a real-time distributed system, due to its NP-hard nature [3]. The large solution space deems linear programming approximations too complex to be feasible for real-time calculation, too. The admission protocol we propose attempts to provide a good quality allocation based on a greedy approach. The protocol computes application utility – the optimization objective – as defined in section 2.1.4. The utility is dependent on the QoS satisfaction ratio, which is proportional to the fraction of allocated resources relative to the difference between the maximum desired value and the minimum. The utility derived from network resources is maximum (1.0) for a connection if both endpoints are mapped on the same node. We also assume that resources for inter-cluster communication are scarce, since traffic may have to pass intermediary hops, and routes may be shared by many connections. Therefore we first attempt to map tasks on the same node, and then, based on resource availability, we map nodes in the local cluster, before going to nodes from remote clusters. This heuristic method attempts to improve application utility by increasing the utility derived from network utility.

3.2.2.2. Minimize access to remote state information

The second design goal for the admission protocol is to use local state as much as possible and to make fewer accesses for information to remote nodes, therefore lowering the protocol overhead. The admission protocol uses network topology information from the routing protocol. One important role in admission is assigned to the cluster head node. The ad-hoc routing protocol determines cluster membership and elects a head for each cluster during route computation. A major task of the cluster head is to forward inter-cluster traffic on behalf of cluster members. Application Service Managers avoid broadcasting requests for resources by forwarding the requests to selected cluster head nodes for local distribution.

3.2.2.3. Adapt to a changing and faulty network

In contrast to a wired network, a MANET is an unstable environment. Nodes move, and noise and jamming degrade link quality. The resource management infrastructure must continue operation in order to support deployed mission-critical applications. The admission protocol consists of two phases. In the first phase ASMs issue reservation requests to RMs and other ASMs. Before the second phase, the initiator ASM computes the greedy allocation and decides a task mapping. During the second phase of the protocol, the initiator ASM commits selected reservations. This two-phase commit protocol, combined with a timeout mechanism, insures that orphan admission requests passed to nodes that have lost connection to the initiator and not yet committed do not block available resources. The two-phase commit protocol is derived from the ripple scheduling protocol implemented in GRMS [7], the difference being the introduction of request timeout and the elimination of the *cancel* operation. This approach allows the protocol to operate without unnecessarily blocking resources or being blocked waiting for requests in case the network looses connectivity, or nodes fail.

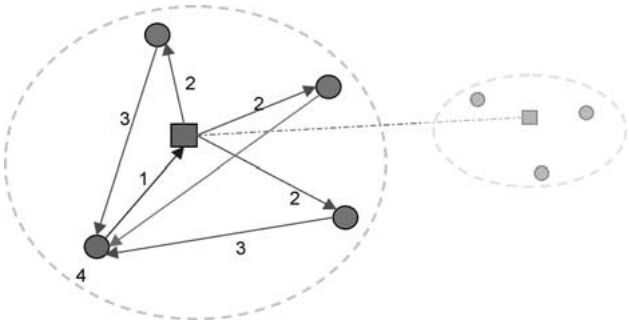

Figure 3. Steps 1–4: local admission.

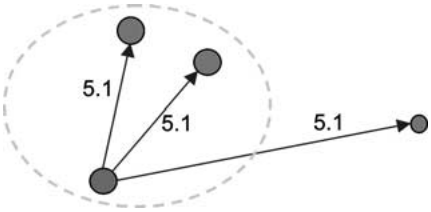

Figure 4. Step 5: admission decision.

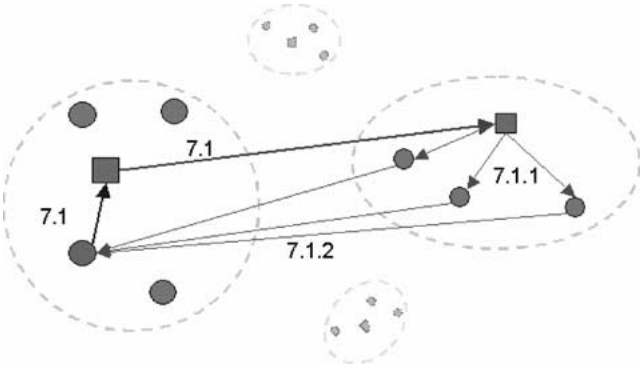

Figure 5. Step 7: remote admission (inter-cluster).

3.2.3. Protocol description

This section gives a pseudo-code description of the admission protocol. Figures 3–5 show the basic steps of the protocol. The square nodes represent cluster heads and the numbers are associated to protocol steps:

0. Client sends admission request to ASM on its node (called "Initiator").
1. Initiator sends admission request to Cluster Head (CH).
2. CH sends (multicast) request to nodes, including priority.

3. Nodes send available node resources r^0, network resources nr^0 as well as $r^{0\prime}$ and $nr^{0\prime}$ to Initiator and Cand, where

$$r^{0\prime} = r^0 + \sum_{i \in \text{Cand}} (r_i^a - r_i^m)$$

and Cand $= \{i \mid p_i < p_{\text{new}}\}$, the set of applications (stages) of lower priority than the new application (similar for all resource types, nr).

4. The initiator performs greedy admission (best/first fit as many tasks on one node).
5. If enough resources,
   5.1. Initiator sends:
      - "Commit" messages to selected nodes.
        Message includes: QoS + app. description + apps to adapt.
      - "Adapt app" to the other nodes hosting stages of adapted apps (if any).
   5.2. Nodes adapt selected applications and admit the new tasks.
   5.3. Return success and allocated contract. STOP.
6. If not enough resources, select applications to adapt. Selection criteria may include priority (absolute), number of tasks, number of adapted applications, app. topology:
   6.1. If resources (available + adapted) are sufficient, GOTO step 4.
   6.2. Else continue with step 7.
7. If not enough resources available:
   7.1. If cluster available:
        initiator asks the CH to forward the admission request to the next CH on the preference list:
      7.1.1. remote cluster head forwards request to its members;
      7.1.2. remote nodes reserve resources and send reply to Initiator;
      7.1.3. GOTO step 4.
   7.2. If no other cluster left
        Initiator returns "unable to admit new application" to Client.
        (All nodes will timeout the initial reservations.)
        STOP.

In section 4 we present simulation results for single and multi-application admission scenarios with topologies with one or more clusters.

4. Simulation results

In this section we present simulation results for the single-application admission scenario and a comparison with application quality approximated with a mixed integer program computed with Matlab, followed by performance results for the multi-application scenario simulations.

4.1. Single-application simulations results

When a node sends an application request to the local ASM, it becomes the centralized point of decision making. First, it tries to admit as many stages of the application as possible on the same node (note that it is possible that none of the stages can be admitted because of lack of resource availability on the node). Then it goes to the next node as provided in its own list/set of preferred nodes to request for admission of the application stages (preference set). The preference set is predetermined and it does not change during simulation time. It then tries subsequent clusters until all stages of the application are admitted or the admission fails.

QoS dimensions
The QoS dimensions we consider are CPU load and network bandwidth. The CPU load is defined by the product of task rate (cycles/second) and workload (work/cycle). The network bandwidth between two tasks is specified by the product of rate and frame size. The frame size and workload are not negotiable. Only the execution rate will be determined and enforced by the system. Both QoS dimensions are specified as a range. The system will determine an acceptable execution rate that will place the CPU load and the network bandwidth within an acceptable range [min, admitted].

Application classes
We use in simulations pipeline applications with 3, 5 and 7 stages and we consider three types of applications: light-cpu-intensive (all stages of the pipeline have light CPU requirements), heavy-cpu-intensive (all stages of the pipeline have heavy CPU requirements) and mixed-cpu-intensive (every stage has variable CPU requirements). We also take into account three classes of resource availability in the network: light load, medium load and heavy load. Thus we have 27 cases: 3 kinds of pipelines × 3 kinds of application CPU requirements × 3 types of system load. The network topology has a single cluster scenario with 10 nodes.

Simulation results
In figure 6, we notice that as the QoS requirements increase, the quality of both the optimum and baseline scheme decreases when the resource availability is high. This is true also

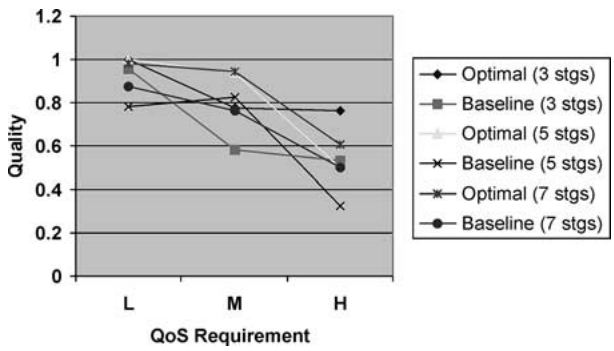

Figure 6. Effect of QoS requirement on admission quality at high resource availability.

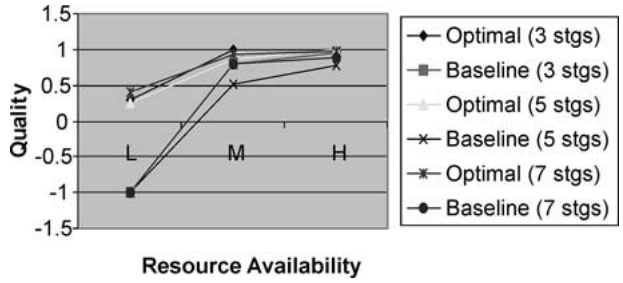

Figure 7. Effect of resource availability on admission quality (with low QoS requirement).

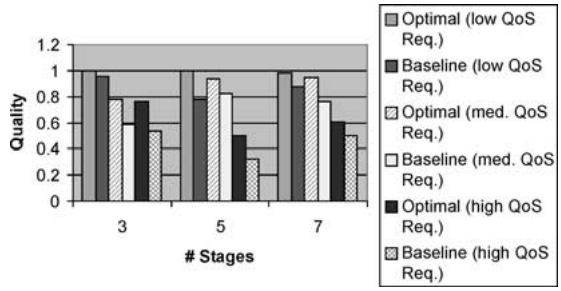

Figure 8. Effect of stage count on admission quality (with high resource availability).

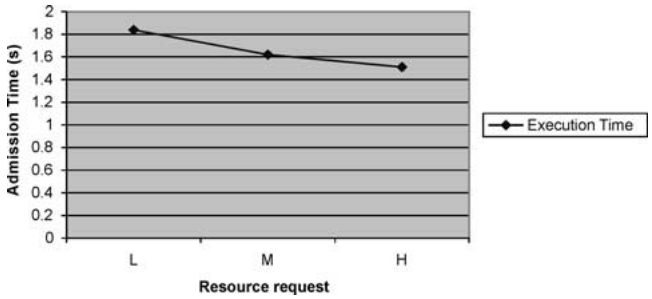

Figure 9. Execution time for a denied admission (7 stages and low resource availability).

when resource availability is low/medium. In figure 7, it can be seen that the quality increases for both schemes, as the resource availability increases, when QoS requirements are low, which is also true for medium/high requirements. A quality value of 1 is the maximum, and −1 indicates admission failure. These observations are expected behavior. In figure 8, we study the effect of the number of stages on quality. We consider the case when the resource availability is high. We would expect that as the number of pipeline stages increases, for a fixed QoS requirement, for instance, low/medium/high the quality should decrease. As seen in the graph, there is really no pattern to this behavior. This behavior seems to be skewed and this is also true for a variety of different runs.

In figures 9 and 10, we study the admission response time (i.e. communication and execution time in Qualnet). Figure 9 illustrates the effect of QoS requirement and resource availability on the response time. We only consider the 7-staged pipeline; the behavior is similar for 3 and 5 stage applications. When resource availability is low, then, as QoS requirements increase, execution time decreases. This happens because the

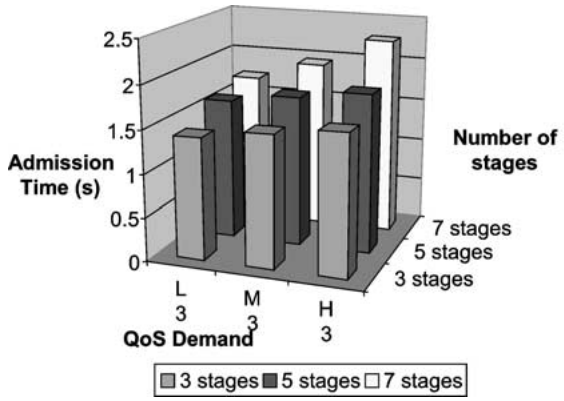

Figure 10. Admission time variation with high resource availability for single-application admission.

greedy algorithm determines faster if the application can be admitted and the likely scenario is for the application to be rejected. On other hand, when resource availability is high, then as QoS requirements increase, then execution time increases. In this case the greedy algorithm needs more time to figure out if the application can be admitted and the likely scenario is for the application to be admitted and more nodes may have to be considered. When resource availability is medium, it is faster to process low QoS requirement (likely admission) and high QoS requirement (likely rejection) than it is to process medium QoS requirement. From these observations, it makes sense to use the "coarse grained" ratio of resource availability to QoS requirements as an indicator for making faster admission decisions, wherein a very low ratio indicates an application to be quickly rejected and a very high ratio indicates an application to be quickly admitted. The slow admission cases are the ones, when the ratio is closer to 1.

The final chart for the single application case in figure 10 shows that when the number of stages increases, the execution time increases, as expected. We only show the case when resource availability is high. This pattern is exhibited similarly for low/medium resource availability.

4.2. Multi-application admission simulation results

For the multi-application simulations we measured the overall admission quality (the admission objective function) and the admission time, depending on the number of applications, resource availability and degree of QoS demands.

The network topology for the multi-application simulation scenarios has 10 nodes grouped in 3 clusters formed at random. Applications arrive at 1 second intervals and both resource availability and QoS requirements are categorized as Low (L), Medium (M) and High (H) as in the single-application case. We analyze the cases where resource demand/availability are H/L, MM and L/H. In our simulations all applications consist of three stages and have the same priority. The system performs no adaptation and each simulation run uses a new set of QoS requests and a new initial state of resource availability. The chart in figure 11 shows the dependency of the overall admission quality (section 2.2.2) on the

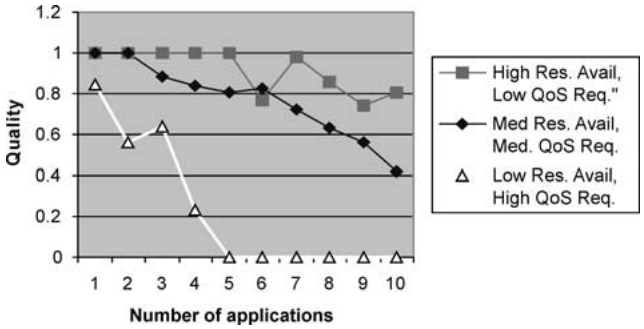

Figure 11. Effect of resource availability, QoS requirements and number of applications on Multi-application admission quality.

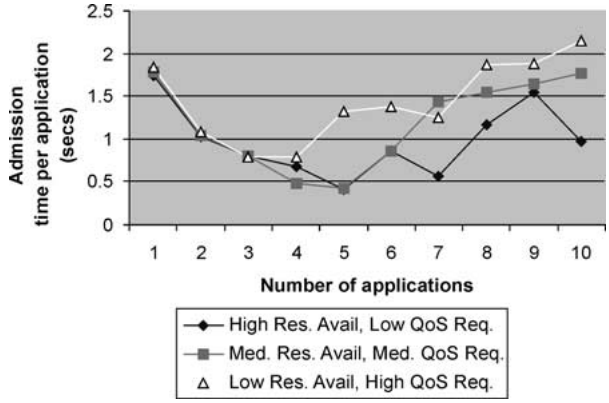

Figure 12. Average admission time in multi-application scenarios.

total number of applications and the degree of QoS demands versus resource availability. We can see that the drop in admission quality is higher when the contrast between resource demands and availability goes up. The system will not be able to satisfy admission requests at the best level when resource availability is low and this will impact the overall admission quality.

Figure 12 illustrates the average admission time for an application depending on the number of applications and the degree of resource demand versus availability. The admission time includes the message communication delay and request processing times. For scenarios with up to 4 applications, the average admission time per application has a similar decreasing evolution for all resource demand/offer combinations. From figure 11 we know that all applications in the 1–4 cases have been admitted. As the number of applications grows, resource availability shrinks, and in the 1–4 application scenarios the search space for each task decreases accordingly. After reaching a minimum at 4 or 5 applications, the communication overhead takes over and the admission time grows back. The longest delays occur for admissions of requests with high QoS demands when resource availability is low. The additional delays are caused by inter-cluster communication when the request initiator has to forward messages to multiple clusters before admitting a new application or giving up.

5. Conclusions

In this paper we presented the resource model, admission protocol and protocol simulation results for the DYNAMIQUE resource management system for MANETs. We defined a multi-dimensional QoS model for the single-application and multi-application admission control, based on QoS satisfaction ratio utility functions. The greedy admission protocol attempts to provide a real-time resource allocation alternative to the mixed-integer programming optimization. The protocol was designed to benefit from network topology information available from the routing layer. In particular, a cluster-based routing protocol, such as CBRP or LANMAR would give Application Service Managers current information on cluster membership and cluster head elections.

The admission protocol targets two areas. First, a heuristic increases application utility by mapping as many tasks on the same node as possible, or on the same cluster, in order to minimize need for wireless communication. The second motivation for cluster organization is to reduce admission control communication costs. The admission protocol supports external adaptation – reducing resources for lower criticality applications in order to admit new applications.

Work on this project is ongoing. The next steps in our research involve simulation evaluation of the multi-application protocol with adaptation, in a mobile scenario, improving the network resource model with end-to-end bandwidth estimation using path information and node transmission capacity, and, in the long run, developing a probabilistic resource model.

Acknowledgements

Research funded by ONR under contract number N00014-01-C-0031. Mihaela Cardei is supported by the University of Minnesota Graduate School Doctoral Dissertation Fellowship.

References

[1] M. Cardei, I. Cardei, R. Jha and A. Pavan, Hierarchical feedback adaptation for real-time sensor-based distributed applications, in: *Proceedings of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing* (March 2000).

[2] I. Cardei, R. Jha, M. Cardei and A. Pavan, Hierarchical architecture for real-time adaptive resource management, in: *Proceedings of the IFIP/ACM Middleware 2000 Conference*.

[3] D.-Z. Du and K. Ko, *Theory of Computational Complexity* (Wiley, New York, 2000).

[4] M. Gerla, X. Hong and G. Pei, Landmark routing for large ad hoc wireless networks, in: *Proceedings of IEEE GLOBECOM 2000*, San Francisco, CA (November 2000).

[5] C. Hoover, J. Hansen, P. Koopman and S. Tamboli, The amaranth framework: Policy-based quality of service management for high assurance computing, International Journal of Reliability, Quality and Safety Engineering, Special Issue on High-Assurance Computing (December 2001).

[6] J. Huang, P.-J. Wan and D.-Z. Du, Criticality and QoS-based multiresource negotiation and adaptation, Real-Time Systems 115 (1998) 249–273.

[7] J. Huang, Y. Wang, N.R. Vaidyanathan and F. Cao, GRMS: A global resource management system for distributed QoS and criticality support, in: *Proceedings of the 4th IEEE International Conference on Multimedia Computing and Systems* (June 1997).
[8] M. Jiang, J. Li and Y.-C. Tay, Cluster Based Routing Protocol, IETF Draft (August 1999) p. 27, http://www.ietf.org/internet-drafts/draft-ietf-manet-cbrp-spec-01.txt
[9] S.-B. Lee and A.-T. Campbell, INSIGNIA: In-band signaling support for QoS in mobile ad hoc networks, in: *Proceedings of 5th International Workshop on Mobile Multimedia Communications (MoMuC'98)*, Berlin, Germany (October 1998).
[10] C. Lee, J. Lehoczky, R. Rajkumar and D. Siewiorek, On quality of service optimization with discrete QoS options, in: *Proceedings of the IEEE Real-Time Technology and Applications Symposium* (June 1999).
[11] C.L. Liu and J.W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of the Association for Computing Machinery 20(1) (1973) 40–61.

Ionut Cardei is a Research Scientist with Honeywell Aerospace Electronic Systems. His main research interests are in ad-hoc wireless networks and distributed resource management. He is currently involved in modeling and simulation of QoS protocols for the DYNAMIQUE project and in designing wireless MAC protocols for coordinated unmanned vehicles. His work on adaptive QoS systems also includes architecture design and development for hierarchical resource management frameworks for distributed real-time applications. Mr. Cardei received a M.S. in computer science in 1999 from the University of Minnesota.
E-mail: Ionut.Cardei@honeywell.com

Srivatsan Varadarajan completed his B.S. from Birla Institute of Technology and Science (BITS), Pilani, India and his M.S. in computer science from University of Minnesota. Currently he is pursuing his M.S. in mathematics and Ph.D. in computer science at University of Minnesota. His interests are in QoS enabled multimedia services over the Internet and mobile ad-hoc wireless environments. He is also a student member of IEEE, ACM and AMA.
E-mail: varadara@cs.umn.edu;
Srivatsan.Varadarajan@honeywell.com

Allalaghatta Pavan received the M.Sc. (Tech.) degree in computer science from the Birla Institute of Technology and Science, Pilani, India, in 1988, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 1998. He has been with Honeywell since 1993 where he is currently a Principal Research Scientist at their Aerospace Electronic Systems Division in Minneapolis. At Honeywell he has been the Principal Investigator and Program Manager for over a dozen research projects in adaptive QoS based middleware systems and in application of multimedia and high-speed networking technologies to hard real-time systems used in process control and industrial automation. His research interests are in QoS protocols, WDM optical networks, wireless ad hoc networks, real-time systems and performance evaluation. He has over 25 publications in these areas. Dr. Pavan was the recipient of the I.B.M. Graduate Fellowship at the University of Minnesota for his Ph.D. thesis work and the National Talent Search Scholarship from the Government of India. Dr. Pavan is a member of the ACM and the IEEE.
E-mail: Allalaghatta.Pavan@honeywell.com

Lee Graba is a research scientist at Honeywell Laboratories. After receiving a B.S. in aerospace engineering from the University of Minnesota, he was employed at Northrop Aircraft, where he designed and analyzed flight control strategies for unstable, highly maneuverable, unmanned, stealthy aircraft. Since receiving an M.S. in electrical engineering from the University of Minnesota, he has worked at Honeywell, first in a group developing industrial control systems, and then moving into the development of distributed, embedded software infrastructures. His recent work has been in adaptive resource management (QUORUM), spontaneous networks (Jini), and middleware for certified avionics software.
E-mail: Lee.Graba@honeywell.com

Mihaela Cardei is a Ph.D. candidate at the Computer Science Department at University of Minnesota. She graduated from the same school with a M.S. degree in 1999. Mihaela is currently supported by the Doctoral Dissertation Fellowship awarded by the University of Minnesota Graduate School. Her current research work is focused on energy efficiency and resource management in ad hoc wireless networks. While working at Honeywell Labs in 1999, Mihaela contributed to the RTARM project new feedback adaptation algorithms for distributed real-time applications.
E-mail: mihaela@cs.umn.edu

Manki Min received B.S. in mathematics and M.S. in computer science from Seoul National University, Seoul, Korea in 1997 and 1999, respectively. He is currently a graduate student in Computer Science and Engineering Department, University of Minnesota. His current research interests are polynomial time approximation schemes (PTAS) and its applications to wireless ad hoc network topologies.
E-mail: min@cs.umn.edu