1-2009

Distributed Utility-Based Rate Adaptation Protocols for Prioritized, Quasi-Elastic Flows

Sharanya ESWARAN
Pennsylvania State University

Matthew P. JOHNSON
City University of New York

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Thomas LA PORTA
Pennsylvania State University

DOI: https://doi.org/10.1145/1558590.1558592

Citation

ESWARAN, Sharanya; JOHNSON, Matthew P.; MISRA, Archan; and LA PORTA, Thomas. Distributed Utility-Based Rate Adaptation Protocols for Prioritized, Quasi-Elastic Flows. (2009). *ACM SIGMOBILE Mobile Computing and Communications Review*. 13, (1), 2-13. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/655

Published in ACM SIGMOBILE Mobile Computing and Communications Review, Volume 13 Issue 1, January 2009, Pages 2-13.
http://doi.org/10.1145/1558590.1558592

Distributed Utility-Based Rate Adaptation Protocols for Prioritized, Quasi-Elastic Flows

Sharanya Eswaran[a] **Matthew P. Johnson**[b] **Archan Misra**[c] **Thomas La Porta**[a]
eswaran@cse.psu.edu *mpjohnson@cuny.edu* *archan@research.telcordia.com* *tlp@cse.psu.edu*
[a]Networking and Security Research Center, Pennsylvania State University
[b]Department of Computer Science, City University of New York
[c]Advanced Technology Solutions, Telcordia Technologies

This paper considers rate adaptation for streaming multimedia data in a wireless sensor network (WSN) consisting of multiple missions, where each mission subscribes to data streams from multiple sensors, and each sensor's data is utilized by multiple heterogenous missions. We specifically consider the application of the distributed network utility maximization (NUM) framework to a previously unconsidered scenario where the different missions have different priorities, as well as minimum utility demands. When all the utility demands are feasible, we first show that the addition of a penalty for failing to reach the minimum utility demand to the base NUM protocol leads to maximization of the global utility. The paper, however, principally focuses on those situations where the minimum demands cannot all be satisfied due to resource constraints. To address such practical scenarios, we present and evaluate a) a family of modified NUM-based protocols to determine the optimal satisfied set, when the missions have unique priority order, and b) heuristics for applying NUM, when multiple missions have the same priority.

I. Introduction

Our work in this paper is motivated by a class of *multi-hop wireless sensor applications* that require the routing of *streaming data* to a group of *tactical* applications that are operational for relatively short time periods (i.e., days rather than months), and that employ sophisticated, high data rate sensors, e.g., video cameras, short-aperture radar and audio sensors. Unlike the majority of WSN scenarios, where sensors are relatively resource-constrained (e.g., Motes) and where sensor data transmission is discrete and event-based (e.g., transmission of motion event alerts by a motion sensor), our focus is on applications (e.g., video monitoring of a site perimeter or trajectory estimation from radar feeds) that consume, and usually perform some signal processing over, *continuous, variable-rate streams* of data from multiple sensors. In such streaming environments, an application's derived utility may be expressed as a function of the received stream data rates. Given the moderately high data rate of each stream being transmitted to multiple receivers (called *missions* in our work) over a multi-hop wireless network, bandwidth (rather than energy) becomes the most critically-constrained resource, necessitating the introduction of *congestion control* algorithms.

The Network Utility Maximization (**NUM**) approach, originally introduced in [1, 2, 7], is an attractive distributed optimization framework for flow-oriented traffic, as it allows an individual user to define its own utility as a function of the traffic rates from its set of sensors. In recent work [3], we extended this framework to embrace two new characteristics of our mission-oriented wireless sensor network (WSN) environment: (i) *Collective Utility*, whereby a mission's utility is defined jointly over multiple sensors, (ii) *Multi-Sink Flows*, whereby each sensor's data may be consumed by multiple missions, each with distinct perceived utilities.

In this paper, we present a distributed solution for utility-based congestion control in a streaming WSN in the presence of *mission priorities*. Accommodating missions of differing priorities is a very important practical requirement, especially in military and emergency response scenarios, where some missions are more critical than others (e.g., gunfire localization vs. perimeter monitoring). The basic NUM framework focuses only on maximizing the cumulative utility, i.e., it is permitted to drive the traffic rates for some missions to very low values, if it contributes to the "collective good"; accordingly, any set of missions is inherently *feasible*.

This paper makes the following key contributions: (i)

We first define a high priority mission via a combination of a *minimum-acceptable* utility demand function and a *priority order*. In general, a mission is modeled to have a quasi-elastic demand. This gives rise to the biggest challenge when introducing prioritized missions: all mission demands may no longer be collectively feasible; (ii) We develop a modified NUM adaptation technique that, under the assumption that all demands are feasible, can meet all minimum demands and maximize global utility. (iii) In more practical scenarios, all the demands may not be feasible together due to bandwidth constraints or policy rules. For the case where all priority missions are strictly ordered, we provide three distributed protocols for determining the optimal subset of missions whose demands can be met and the maximum utility that can be achieved. (iv) When multiple missions are collectively infeasible and have the same priority, we show that the NUM framework is insufficient for solving the problem, and present a set of heuristics for dealing with this scenario.

We start with the assumption that all minimum demands of prioritized missions are feasible, using this scenario to introduce the formal definition of prioritized missions and to establish the basic changes needed to the NUM framework to achieve optimal rate adaptation, while considering both the elastic and inelastic demand components. The bulk of our work then involves the use of this basic framework to address the more complex and realistic cases in which the combined minimum demands of prioritized missions are not feasible.

The rest of this paper is organized as follows: Section II provides a brief overview of related work. In Section III we present the modified mathematical model of optimization for priortized missions with quasi-elastic demands, when all demands are feasible. In Section IV, we describe and quantitatively evaluate three protocols for selecting the optimal subset of demands, when not all are feasible. In Section V, we evaluate heuristics for the case where missions have non-unique priorities. Finally, Section VI concludes the paper.

II. Related Work

The classical NUM framework ([1, 2, 7]) was recently extended in [3] to a more general WSN environment, where the topology is arbitrary, individual sensors have multiple subscribing missions, individual missions derive their utility from a composite set of sensors, and intermediate nodes use link-layer multicast to forward sensor data downstream; this may be viewed as a generalization of NUM for multi-rate, wireless multicast performed in [14].

In this WSN-centric model, the optimization problem is formulated as: $SENSOR(U, L)$:

$$\textbf{maximize} \sum_{m \in M} U_m(X_m) \ \textbf{subject to} \sum_{\forall (k,s) \in l} \frac{x_s}{c_{k,s}} \leq 1,$$

where $U_m(X_m)$ represents the utility function of mission m (M being the set of all missions) as a function of the S-dimensional vector of rates associated with the set of sensors S; $c_{k,s}$ is the transmission rate used by node k during the link-layer broadcast of the data from sensor s, and L is the set of all maximal cliques in the conflict graph (CG) corresponding to WSN; please see [3] for details. Based on this new model, a sensor (source) $s1$ adapts its rate as: $\frac{d}{dt} x_{s1}(t) =$

$$\kappa\Big(\sum_{m \in Miss(s1)} w_{ms1}(t) - x_{s1}(t) \sum_{\forall l \in flow(s1)} \sum_{\forall (k,s1) \in l} \frac{\mu_l(t)}{c_{k,s1}} \Big) \quad (1)$$

where $\mu_l(t)$ is the 'cost' charged per bit by each forwarding clique, and is given by $\mu_l(t) =$

$$p_l\Big(\sum_{\forall (k,s) \in l} \frac{x_s(t)}{c_{k,s}} \Big) = \Big(\sum_{\forall (k,s) \in l} \frac{x_s(t)}{c_{k,s}} - 1 + \varepsilon \Big)^+ / \varepsilon^2 \quad (2)$$

Each mission (sink) adapts its 'willingness to pay' terms w_{ms} (these terms represent pure Lagrangian variables; there is no actual pricing or monetary exchange) for sensor s based on the source rates and its own utility function $U_m(.)$, according to the equation $w_{ms}(t) = x_s(t) \frac{\partial U_m}{\partial x_s}$. The system converges to an optimal solution of a relaxation of the problem $SENSOR(U, L)$.

Past work on priorities in NUM-based optimization has been done mostly in the context of admission control and differentiated services. In [6] and [5], different 'shapes' of utility functions are considered for different classes of flows. A control-theoretic approach for QoS provisioning in ad hoc wireless networks is specified in [8], where the rates of the low-priority flows are changed such that the high-priority flow is maintained at the required rate. Unlike these approaches, we focus on techniques that are *independent* of specific choices of utility functions.

In [9], the NUM framework is used for providing differentiated QoS-based rate allocation, where each class of user defines a minimum and maximum allowable rate. In [10], the NUM model is analyzed for inelastic and non-concave utility functions. While we explicitly distinguish between a mission's priority order and its minimum demand to help arbitrate cases where all demands cannot be accommodated, prior approaches, including [9] and [10], do not consider explicit priorities - a user's importance is simply captured by its demand.

III. The Prioritized-NUM Framework (All Demands Assumed Feasible)

In this section we show how the WSN-NUM model in [3] can be enhanced to provide differentiated rate control based on the minimum utility demands of high priority missions. We initially focus on the simple case where all demands are feasible; so, we do not yet explicitly consider the priority order of missions. The main objective of this section is to define the notion of demands within the NUM-framework and to show how both quasi-elastic and elastic (best-effort) flows can be accommodated together.

III.A. Prioritized Flows as Minimum Demand Constraints

Without loss of generality, missions can be categorized into 'regular' and 'prioritized' missions. Regular missions do not possess any minimum-utility demands. In contrast, each prioritized mission m has a *minimum utility requirement* (expressed as a function over the rates of each of the sensors $s \in set(m)$) that must be met if resources are available. Prioritized missions are also quasi-elastic and would like to utilize additional bandwidth (*i.e.*, have more sensors transmit at higher rates) if the wireless network is not congested and additional capacity is available.

The definition of a mission's utility as a *collective* function of multiple sensors implies that its minimum requirement can conceivably be specified as one of multiple alternative combinations of rates from $set(m)$. As a generalization, prioritized missions specify their minimal utility requirements as a *demand function that can be represented as:* $f_i(X_m) \geq D_i$. This effectively specifies an l-dimensional surface in the utility space, demarcating the 'unsatisfied' region. (Here, l is the number of sources that the demand involves.) Thus, a source node that serves a high priority mission has partially-elastic flows, where its rate can be adjusted as long as it is within the feasible region.

III.B. Enhanced NUM framework with minimal demands

The optimization problem, with the demand functions of high priority missions represented as additional constraints, is:

$SENSOR - P(U, L)$:

$$\begin{aligned} &\textbf{maximize} && \textstyle\sum_{m \in M} U_m(X_m) \textbf{ over } x_s \geq 0 \\ &\textbf{subject to } \text{i)} && \textstyle\sum_{\forall (k,s) \in l} \frac{x_s}{c_{k,s}} \leq 1, \textbf{ and} \\ & \text{ii)} && f_i(X) \geq D_i, \ \forall i \in \{1, \ldots, H\} \end{aligned} \quad (3)$$

where $f_i(X)$ denotes the minimum demand of a prioritized mission in terms of its utility, and H is the total number of such 'prioritized' missions. This can be solved by decomposing into independent $SINK$ and $NETWORK$ problems.

$SINK_m(U_m; \lambda_m)$:

$$\textbf{maximize } U_m\left(\frac{\bar{w_m}}{\bar{\lambda_m}}\right) - \left(\sum_{s \in set(m)} w_{ms}\right) \textbf{ over } w_{ms} > 0 \quad (4)$$

where $\bar{w_m}$ is a vector of terms w_{ms}, $\bar{\lambda_m}$ is a vector of λ_{ms}, and element-wise division of $\bar{w_m}$ by $\bar{\lambda_m}$ is assumed.

$NETWORK(L; w)$:

$$\begin{aligned} &\textbf{maximize} && \textstyle\sum_{s \in S} \sum_{m \in M} w_{ms} log(x_s) \textbf{ over } x_s \geq 0 \\ &\textbf{subject to } \text{i)} && \textstyle\sum_{\forall (k,s) \in l} \frac{x_s}{c_{k,s}} \leq 1, \text{ for each clique } l \in L, \\ &\textbf{and } \text{ii)} && f_i(X) \geq D_i, \ \forall i \in \{1, \ldots, H\} \end{aligned} \quad (5)$$

We can convert this to an unconstrained problem using Lagrangian multipliers on Eq. (5). Based on the first-order necessary condition for optimality, a sensor $s1$ adjusts its transmission rate according to the differential equation:

$$\frac{d}{dt} x_s(t) = \kappa\Big(\sum_{m \in Miss(s)} w_{ms}(t) - x_s(t) * \left(\sum_{\forall l \in flow(s)} (\mu_l(t) \sum_{\forall (k,s) \in l} \frac{1}{c_{k,s}}) - \sum_i (\eta_{\mathbf{is}} \frac{\partial \mathbf{f_i(X)}}{\partial \mathbf{x_s}})\right)\Big) \quad (6)$$

where μ_l and η are the shadow costs, given by Eq. (2) and Eq. (7) respectively.The above equation is similar to the basic adaptation algorithm (Eq. (1)), except for the additional terms involving η_i. η_i can be viewed as the per-bit cost of not adhering to the demand; this term ensures that the overall 'willingness to pay' indicated to a sensor $s1$ is higher than the sum of the marginal utilities of $Miss(s)$ by an amount proportional to the distance of the rate from an appropriate 'feasible point' (one that satisfies the demand function) if the demand is currently unmet. In general, if a demand i is given by $f_i(X) \geq D_i$, then the normalized η_{is} for sensor x_s at time t can be given by the following function:

$$\eta_{is} = \{(1 - f_i(X(t))/D_i)^+/\epsilon^p \text{ if } f_i(X) < D_i; \text{ =0 otherwise } \} \quad (7)$$

where $0 < \epsilon < 1$ and ϵ plays a similar role as ε in Eq. (2); p is the exponent of the denominator that determines the magnitude of the cost (per unit flow) of not adhering to the minimum demand.

Under the assumption that all demands can be met, we can prove that the system converges at the maximum global utility with penalties for congestion and for not meeting the minimum demands. (The proof has been omitted due to space constraints, and is provided in [13]). But this does not guarantee that all demands will be satisfied, because it is possible that the global optimum utility of some missions may offset the cost of not meeting the minimum demand of some other missions. To avoid this, we must ensure that *a)* a 'happy' mission can never overwhelm even the least 'unhappy' mission, and *b)* the marginal increase in utility of *all other missions* must be lower than the marginal decrease in dissatisfaction for any currently-unsatisfied mission.

The first condition is met by ensuring that the maximum slope of the global utility (due to concavity of utilities, this occurs at the point $x_s = 0$ for elastic missions) must always be less than the η-function. For a tractable solution that satisfies the second condition, we fix an upper bound, N_{max}, on the number of missions that the system can admit, and also restrict the maximum slope of any mission's utility function to S_{max}. According to Eq. 7, if the current utility is within a bound of ϵ from the demand curve, *i.e.,* if $f_i(X)/D_i(X) = 1 - \epsilon$, then $\eta_i = \frac{1}{\epsilon^{p-1}}$. Hence, we can guarantee that if the demands are collectively feasible, then they will all be met (to within ϵ) provided the following condition holds: $\frac{1}{\epsilon^{p-1}} > N_{max} * S_{max}$.

Intuitively, this condition ensures that the marginal cost for a prioritized flow receiving utility that is more than ϵ lower than its minimum demand is always greater than the marginal utility of all other missions.

From a practical protocol-level perspective, the only change in the NUM protocol occurs at a receiver (mission), which now sends an η-corrected path cost to the source, *i.e.,* the new formulation does not incur any additional overhead. Moreover, a high priority mission computes its 'willingness to pay', based on the source rates and its own utility function $U_m(.)$, according to the modified equation:

$$w_{ms}(t) = x_s(t)\frac{\partial U_m}{\partial x_s} + x_s(t) * \eta_{ms} * \left(\frac{\partial f_m(X)}{\partial x_s}\right). \quad (8)$$

IV. Prioritized-NUM With Infeasible Demands and Strict Priorities

The basic prioritized NUM framework, discussed in the previous section, cannot be applied directly when the utility demands cannot all be satisfied together, due to capacity constraints or policy rules. The issue of utility maximization under *infeasible* constraints is, in fact, absent from prior work on NUM-based rate control (irrespective of whether flows were considered to be completely elastic or inelastic). In this section, we consider the case where the different quasi-elastic missions (defined by the set $HP = \{\text{all priority flows}\}$) have a *priority order (PO)* assigned to them. A higher value of PO implies higher priority; furthermore, missions have a *strict priority ordering* (no two missions have the same PO).

Given this setting, the problem is to find the "optimal set" of 'satisfied flows' SF ($SF \subset HP : m_i \in SF$ iff $f_i(X) > D_i$), where D_i is the demand constraint for mission m_i.

Optimality is defined by the following properties: (i) *Priority Property*: A mission m_i with priority PO_i cannot be in the set SF if removing this mission from SF (i.e., reducing its utility below its minimum demand) enables a higher priority mission, $P_j : P_j > P_i$, not in set SF, to become a member of SF, and (ii) *Utility Property*: Given the set SF, the set of sensor rates chosen, $\{x_s\}$, maximizes system utility (subject to capacity constraints) and meets the minimum demand for all $m_i \in SF$. If $m_i \notin SF$, it is simply treated as a regular mission (with no minimum demands) and thus still receives a proportionally-fair data rate.

This generic problem cannot be solved using the gradient-based NUM approach directly; an additional step is required to compute the optimal set SF. Intuitively, a lower priority (higher PO) mission is admissible only if its demand can be met within the "feasible region", in the sensor rate space, defined by higher priority demands. We describe and evaluate three different *fully-distributed* protocols: (i) *Incremental*, (ii) *Batch* and (iii) *Hybrid*, that perform this feasibility evaluation.

IV.A. Incremental Protocol

In this protocol, we start by ignoring all demand constraints and iteratively add constraints in the decreasing order of their PO, *i.e.,* adding the highest priority demand first. At each iteration, as long as the demands of all the previously added (higher priority) missions are satisfied, the newest mission remains in the set SF; otherwise it is dropped (excluded from SF). The feasibility for any given set, SF, of demands is evaluated by running the NUM algorithm described in Sec. III and verifying that the network converges at a point where all demands are met. When a mission m's demand i is 'added' to the system, the mission sends its sources the feedback term η_{is} (Eq. 7). If it has not been included yet, or has been dropped, the mission sends $\eta_{is} = 0$.

This protocol requires missions to identify the highest priority unsatisfied mission in the network at any instant. This is achieved by modestly extending the basic protocols, described in [3] and Sec III, by having the nodes maintain and propagate additional book-keeping information. During the construction of the local conflict graph, nodes exchange information about the set of flows that they transmit, as well as the priorities of the receivers (missions) that consume the flows traversing the node. At the end of the local conflict graph construction, each node is, thus, aware of the highest priority mission in that clique, defined as the *one with the highest PO value among all missions that consume data from a node involved in the clique.*

In the basic clique-based congestion framework of [3], the nodes within a clique communicate periodically to exchange the current air-time fractions (which is used to compute the clique cost). During these periodic exchanges, each node now also exchanges the highest-priority information, enabling the identity of the mission with the highest priority value in the clique to percolate to all clique members.

When a node transmits data, it also appends the highest priority seen by any of its cliques. When a node receives the packet, it checks if its own clique is aware of a higher priority mission; if so, it updates the packet. By the time the packet reaches its destination, it contains information about the highest priority mission. The mission checks if it is the highest priority mission-if so it then has the 'right of way' to signal its utility demand. This mission transmits the additional feedback term, η as computed by Eq.7, along with the congestion cost feedback. If the demand is feasible, it is met, the rates converge and the next highest priority mission can get its turn. If it is not feasible, then either its demand is not met or it makes a higher priority mission unsatisfied. If this happens, then this mission is "rejected" and must not reattempt to signal a non-zero η value.

To ensure this behavior, each mission maintains a *state*, corresponding to the state diagram in Fig. 1. The solid lines indicate normal transitions and the dotted lines indicate transitions that occur due to system dynamics, *i.e.,* if a mission leaves or enters the network. Initially, a mission is in *Candidate* state, when its demand has not yet been tried. When it gets its right of way, its state changes to *Active*. After the rates converge, an Active mission transitions to either *Rejected* or *Satisfied*. A *Satisfied* mission is *Violated* if a lower priority mission's demand makes it unsatisfied.

The current state of a mission is sent (as a concise bit vector) along with its feedback message to the source sensors, which then piggy-back this information on the data packets transmitted on the forward path; intermediate nodes overhear these transmissions and the propagate this state information along all their flows. This way, each forwarding node can pick the highest unsatisfied *Candidate* mission, and the current *Active* mission can infer whether its demand has made a higher priority mission unsatisfied (in which case it moves to the *Rejected* state). We note that both the selection and reject decisions are made *locally* by each mission, without any external arbitration.

Adapting to Dynamics: The protocol must also adapt to the arrival of a new mission or the departure of an existing mission. The basic NUM protocol without priorities can itself be adapted to such dynamics as described in [3]. If the priority of the new mission (or exiting mission) is lower than that of the currently *Active* mission, then it simply waits for its turn. If the new (or exiting) mission's priority is higher, then the currently *Active* mission and all missions with priority lower than the new (or exiting) mission must revert back to being *Candidate* and reset their η to 0.The arrival of a new mission is *implicitly* captured by the protocol, as the cliques have been updated with the new mission's information during their reconstruction. However, when a mission exits the network, and it is not a *Candidate*, then explicit "Reset" messages must be propagated over the WSN to trigger the lower priority missions to revert to the *Candidate* state; this trigger is initiated by the sensors upon prolonged absence of feedback from a subscribed mission.

IV.B. Batch Protocol

In contrast to the Incremental protocol, the Batch protocol has all 'prioritized' missions try their demands in parallel. If there is at least one unsatisfied demand after the rates converge, we remove the demand constraints successively

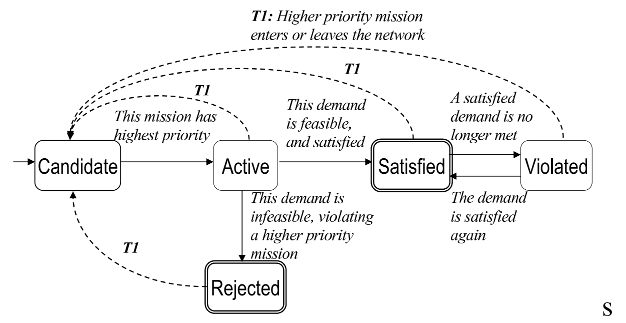
T1: Higher priority mission enters or leaves the network
T1
T1
This mission has highest priority
This demand is feasible, and satisfied
A satisfied demand is no longer met
Candidate
Active
Satisfied
Violated
This demand is infeasible, violating a higher priority mission
The demand is satisfied again
T1
Rejected
s

Figure 1: State Transition Diagram for a mission in Incremental Protocol

in the increasing order of priority, *i.e.,* removing the least priority demand first. This process defines a *drop-phase* and continues until sufficient missions are dropped to ensure that the demands of all the remaining higher priority missions are met. But this alone does not result in an optimal set SF, because some of the lower priority demands that were dropped may possibly be rehabilitated. Hence, the protocol has a second *restore-phase*, where the dropped demands are retried sequentially (similar to the Incremental protocol), in descending order of priorities.

The implementation of this protocol is similar to the Incremental protocol. The lowest priority missions, during the 'drop' phase, and the highest priority dropped missions, during the 'restore' phase, are detected as explained in IV.A. The state diagram of a mission is shown in Fig. 2. All missions are initially in the *Active* state. If its demand is not met, its state changes to *Violated* which triggers the 'drop-phase'. A mission being *'Tried'* in the restore-phase becomes either *'Active'* or *'Rejected.*

Adapting to Dynamics: When a new mission of priority PO_i arrives during the drop-phase, and its priority is lower than the mission dropped previously, then it is *Dropped*, otherwise, it remains *Active*. If it arrives during the 'restore' phase and has $PO_i < PO_j$, where j is the mission currently being *Tried*, it changes state to *Dropped* and waits for its turn to be *Tried*. Otherwise, the new mission and all missions with $PO < PO_i$ enter the *Dropped* state, implicitly restarting the 'restore' phase. If the new mission arrives after existing missions have converged, then it enters the *Active* pool, and the protocol proceeds as before. If a mission leaves during the 'drop' phase or a *Dropped* or *Tried* mission leaves during the 'restore' phase, it does not alter the behavior of the remaining missions. In all other cases, missions with PO values lower than the exiting mission are *Dropped* and 'restore' phase is (implicitly) restarted.

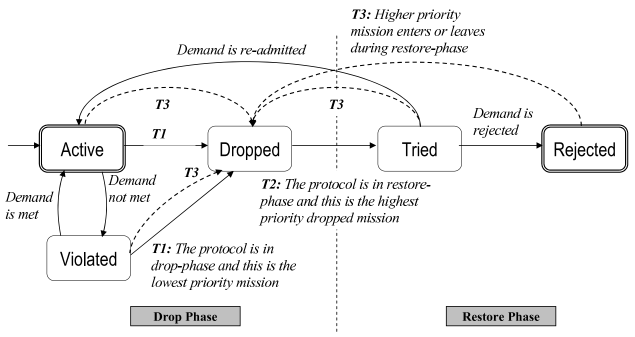
T3: Higher priority mission enters or leaves during restore-phase
Demand is re-admitted
T3
T3
T1
Active
Dropped
Tried
Rejected
Demand is rejected
Demand is met
Demand not met
T3
T2: The protocol is in restore-phase and this is the highest priority dropped mission
Violated
T1: The protocol is in drop-phase and this is the lowest priority mission
Drop Phase
Restore Phase

Figure 2: State Transition Diagram for a mission in Batch Protocol

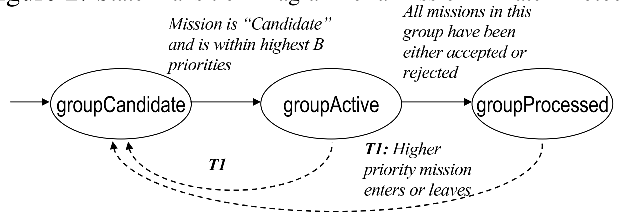
Mission is "Candidate" and is within highest B priorities
All missions in this group have been either accepted or rejected
groupCandidate
groupActive
groupProcessed
T1
T1: Higher priority mission enters or leaves

Figure 3: State Transition Diagram for a mission in Hybrid Protocol

IV.C. Hybrid Protocol

The Hybrid protocol combines the Incremental and Batch protocols: the Batch protocol is run with a group (batch) of B missions, with the Incremental protocol being applied across the batches. For example, with $B = 2$, we try meeting two demands at a time, starting with the highest two priorities. If both are feasible, we move on to the next set of two (i.e., incrementally to the next batch). If any entire group of prioritized missions proves infeasible, the two-phase conflict resolution process of the Batch protocol is invoked within this group. We can see that the Incremental and Batch protocols are special cases of this protocol when $B = 1$ and $B = |M|$, respectively.

The working of the Batch protocol is essentially a combination of the previous two protocols. In addition to the Batch state-machine, each mission also maintains a state-machine for its group, as shown in Fig. 3, whose initial state is of *groupCandidate*. When a mission receives a packet and finds itself as one of the highest B missions, its GroupState value is *groupActive* and its local state becomes *Active*, indicating that it is part of the B missions currently being evaluated. After the batch protocol for this group of B missions successfully terminates (with a subset of these missions potentially being *Rejected*), all the B missions change their GroupState value to *groupProcessed*, indicating that the next batch of B missions can now be evaluated.

Adapting to Dynamics: This protocol deals with the dynamic arrival or departure of missions in a simple way. If a new mission arrives and has a priority PO_{new} that lies within the priority range of the batch B currently being evaluated, i.e., $\min_{i \in B} PO_i < PO_{new} < \max_{i \in B} PO_i$, then the set of B missions is reset to *Active* and the new batch of B missions (containing the new mission) is re-initiated. If $PO_{new} > \max_{i \in B} PO_i$, then the protocol reinitiates the Batch mode by resetting to *Active* all missions that had been previously evaluated and have PO values lower than PO_{new}. The protocol is unaffected if PO_{new} is lower than any of the priorities currently being evaluated. A similar adaptation occurs whenever a mission terminates and leaves the network.

IV.D. Distributed Implementation of the Protocols

In Sections IV.A, IV.B and IV.C, we described how the highest or lowest priority missions are detected in a distributed fashion and how the state information is propagated across the network. In this section, we describe in further detail, the protocol behavior at the source, forwarding node and sink nodes. Figures 4,5 and 6 provide the pseudocode of the algorithm implemented by sensors, forwarding nodes and sinks, for the Incremental protocol. We omit the detailed pseudocode for the other two protocols for reasons of space.

In both Batch and Hybrid protocols, the source and forwarding nodes behave as in Incremental, except that, in Batch, the lowest priority mission in the clique is computed, and in Hybrid, the highest priority B missions are computed. The sink behavior of both protocols is guided by their corresponding state transition diagrams. All the three protocols converge in completely decentralized fashion, even in the face of *transient inconsistencies* that might occur while forwarding nodes exchange information regarding their downstream missions.

IV.E. Performance Evaluation of The Three Protocols

To quantitatively evaluate the performance of the three protocols, we simulated their behavior, using the Qualnet [4] discrete-event simulator, on an 802.11-based wireless network. The network consists of 100 nodes that are randomly deployed on a 1500m x 1500m field. There are 15 flows and 10 missions (i.e.,$M = 10$), of which 5 are prioritized and the rest are regular.

The evolution of network utility over time for Incremental, Batch, and Hybrid-2 (*i.e.,* Hybrid with $B = 2$) are shown in Fig. 7. It also shows that the protocols adapt to dynamics robustly, where the following case was simulated: Initially network has missions with priorities

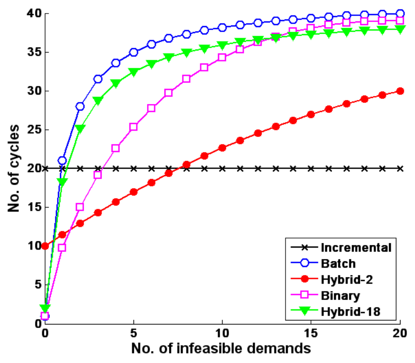

Figure 11: Average convergence time.

1,2,3,4,5. At $t = 150s$, the priority 2 and priority 3 missions leave; at $t = 175s$, a new mission enters with priority 3. We see that by $150s$ all protocols reach the same convergence point. We see that when the new mission enters, Batch approaches the optimal utility faster: while Batch immediately admits the new mission, the Incremental (and Hybrid) must reset the states of lower priority missions and process the demands sequentially (and quasi-sequentially).

Fig. 7 also shows that the protocols have different convergence times. We note that the relative convergence times of the three protocols depend on a number of factors - the demanded utility values, the capacity of the network, number and priorities of demands that eventually turn out to be infeasible, and in Hybrid protocol, the specific group size B used. Fig. 8 shows the mean convergence time (and the 95% confidence intervals) taken by Incremental, Batch and Hybrid (with 3 different group sizes) protocols, for a set of demands with all possible priority arrangements. For this set of demands, where the last two or three demands were always infeasible, we see that Batch converges faster than Incremental, while the performance of Hybrid depends on the group size (and the number of infeasible demands).

Optimal Batch Size Selection for Hybrid Protocol:

As shown above, the performance of the Hybrid protocol greatly depends on the group size. This gives rise to the interesting problem of determining the optimal batch size B that consumes the least number of cycles (n), where a cycle is defined as the process of running the NUM protocol and allowing it to converge, and based on the converged value, deciding if a mission's demand has been accepted or not. If p_i is the conditional probability that the demand of mission i (*i.e.,* with priority i) is feasible given the current network capacity, then the number of cycles required by the Hybrid protocol for a group size

```
Input:
(i) For each trx in its Local Conflict Graph(LCG):trx node ID,
flow ID, airtime fraction ($\frac{x_s}{C_s}$),(priorities,cur. state) of sinks;
(ii) Feedback message={aggregate flow cost, destination state}.
Variable: STATE: state vector for all its missions; RATE: src rate

onReceiveFeedback (destination d): {
        $RATE \leftarrow RATE + computeDx()$
        ∀ trx $T \in LCG$where $T.destination = d$
        update(STATE)
}
onSendPacket(void) {
        $maxP \leftarrow computeMaxPriority(clique)$
        $cliqueCost \leftarrow computeCliqueCost()$
        Transmit(data,cliqueCost,cliqueID,maxP,STATE)
}
computeDX(): {returns result of Eq.6}
computeMaxPriority(): {returns highest priority
seen in the clique}
computeCliqueCost(): {returns result of Equation2}
```

Figure 4: Distributed Adaptation Algorithm at Sensor (Source)

```
Input: For each trx. in its LCG, trx:
nodeID,flowID, $\frac{x_s}{C_s}$,($PO$, cur. states)
for all sinks of the flow }

onReceiveData (Flow f, Data data) {
∀ trx. $T \in LCG$
such that $T.destination \in f.destination$,
  update destination state from
  data.STATE;
if ($nodeId \in data.immediateRx$) {
//i.e., node is on the flow's
// forwarding tree
  $data.maxP =$
$computeMaxPriority(data.maxP \bigcup clique)$
  update data.STATE from
  T.destination state
}
$Data.cost = data.cost+$
$computeCliqueCost()$
}
```

Figure 5: Priority Arbitration Algorithm at Forwarding Node

```
onReceiveData(flow f) {
if ($State! = “Candidate'' \ AND \ (data.maxPri >_{pri} myPri || data.exitedMission.Pri >_{pri} myPri)$ {//new mission arrives
State = ‘‘Candidate’’;
} if ($State! =''\ Satisfied'' \ AND \ State =''\ Rejected''$) {/* a decision has not yet been made on this mission*/
    If( data.maxPriority = myPriority){ /* this mission has the current highest priority */
        $\eta \leftarrow computeUnsatisfiedCost()$
        If($''Violated'' \in data.STATE\&\&isConsistent()$) then $State = “Rejected''$
        else if ($\eta = 0$ && $isConsistent()$) then $State = “Satisfied''$
    } Else $\eta \leftarrow 0$
} else if(State = ‘‘Satisfied’’) { /* this mission has higher priority than the current highest,
and its demand was feasible with its predecessors and hence was admitted to the system */
    $\eta \leftarrow computeUnsatisfiedCost()$; //0 if the demand is still met
If($\eta > 0$) $State = “Violated''$;
} else $\eta \leftarrow 0$ /* if demand is not highest priority nor has been rejected, set $\eta$ to 0 */
w = computeWillingnessToPay(); sendFeedback($\eta$, State, w, data.cost);
}
computeUnsatisfiedCost(): {returns result of Eq.7}
```

Figure 6: η-Adjustment Algorithm at Sink (Mission)

B, is given by

$$n = \sum_{g=1}^{\lfloor \frac{M}{B} \rfloor}[1 + (1 - \prod_{i=(g-1)B+1}^{(g-1)B+B} p_i)\{2(1 - p_{(g-1)B+1})* (B-1) + \sum_{i=(g-1)B+2}^{(g-1)B+B} (1-p_i)(\prod_{j=(g-1)B+1}^{i-1} p_j)* (1 + 2gB - 2i)\}] + [1 + (1 - \prod_{i=\lfloor \frac{M}{B} \rfloor B+1}^{M} p_i)* \{2(1 - p_{\lfloor \frac{M}{B} \rfloor B+1})(M \bmod B - 1) + \sum_{i=\lfloor \frac{M}{B} \rfloor B+2}^{M} (1-p_i) (\prod_{j=\lfloor \frac{M}{B} \rfloor B+1}^{i-1} p_j)(1 + 2M - 2i)\}] \quad (9)$$

The optimal batch size is that value of B that minimizes Eq. 9. However, it is not straightforward to compute the optimal batch size, because it requires knowledge of the

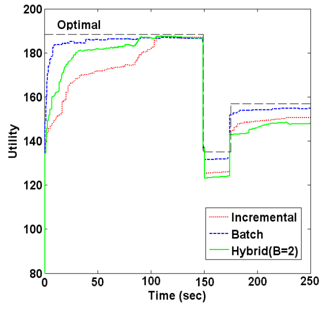

Figure 7: Transient network utility, with adaptation to dynamics.

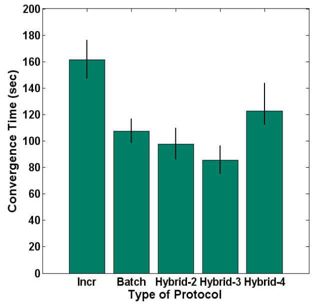

Figure 8: Convergence times for the protocols.

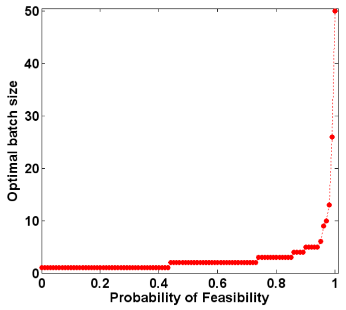

Figure 9: Optimal batch sizes for uniform prob. function, with $M = 50$.

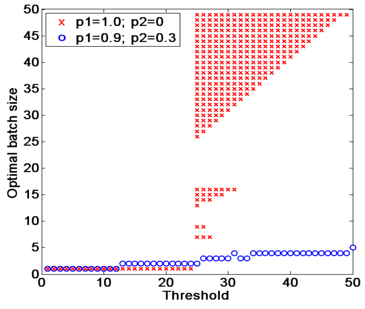

Figure 10: Optimal batch sizes for step prob. function, with $M = 50$.

conditional probability that a demand will be accepted, which depends on factors such as capacity of the network, presence of other missions, priority of demands, and so on. Given the unlikelihood of having all of this knowledge, we turn to finding a reasonable guideline for setting a batch size that is robust in terms of performance. We thus plot the batch sizes that provide the best performance for a variety of probability distributions (for feasibility of mission demands) in Figures 9 and 10.

Fig. 9 shows the optimal batch sizes in a network with 50 missions ($M = 50$) when all demands have equal probability of being accepted. This is often the case when smaller demands are likely to be accepted and larger ones are likely to rejected, and a demand has an equal chance of belonging to either of these categories. For a $probability, p < 0.5$, batch size of 1 is the best (Incremental protocol). If $p \approx 1$, batch size of M is the best (Batch protocol). Otherwise, a batch size of 2-3 is good. Interestingly, we observed the same trend for other values of M too, implying that the best value of B is independent of M.

Sometimes, the demands may all be more or less equal, and the network can accommodate as many demands as the capacity allows. In such cases, it is likely that when a demand of a certain priority is rejected, all demands of lower priority are rejected also. This probability distribution may be modeled as a step-function where, for instance, the probability of acceptance for the first K missions is $p_1 = 1$ and for the remaining $M - K$ it is $p_2 = 0$. Fig. 10 shows optimal batch sizes when the threshold K is varied, for two cases: (i) $p_1 = 1, p_2 = 0$ and (ii) $p_1 = 0.9, p_2 = 0.3$, with $M = 50$. We see that when more demands are certain to be accepted, larger batch sizes are optimal; otherwise, smaller batch sizes are better. Also, when K is greater than and close to the midpoint, a range of batch sizes are optimal. (Indeed, for case(i), choosing a batch size in the range $[\lceil\frac{M}{2}\rceil, M - 1]$ always results in $2 + (M - K)$ cycles, one for each batch, and M-K incremental attempts for each infeasible mission). However in practice, it is better to choose the smallest batch size in the range for two reasons: (i) The penalty of selecting a non-optimal batch size is lower for smaller batch sizes, and (ii) We observed in our experiments that, as anticipated, the duration of a "cycle" is longer for bigger batch sizes.

If there is no a priori knowledge of the demands, then it is safe to assume that the higher priority demands have better chances of being accepted because they are tried and admitted first. We modeled this as a linear probability function of priority (e.g., $p(i) = (M - i)/M$), and computed the optimal batch size. Interestingly, the optimal batch size always turns out to be 1, implying that Incremental-mode is the best for this case.

We thus observe that the optimal batch sizes depend heavily on the conditional probability distribution. In Fig. 11, we plot the convergence time for Hybrid, Incremental and Batch protocols and a binary search algorithm, varying the number of infeasible demands, for $M = 20$, when all missions have the same probability of acceptance. (Binary search is an intuitive approach to mission grouping and is included for comparison). From Fig. 9, we select $B = 2$ as our best 'choice' for the Hybrid protocol. The convergence time is averaged over all possible priority assignments for the infeasible demands and hence is independent of the specific demands (unlike Fig. 8). We see that, in most cases, when fewer demands are infeasible, Hybrid-2 performs better than the rest and otherwise, while it does not perform as well as the Incremental, it does much better than the other two algorithms. We also observe that Hybrid-18 performs much worse than Hybrid-2, implying a large penalty for choosing an incorrect large batch size.

V. Prioritized-NUM with Infeasible Missions and Non-unique Priorities

In this section, we relax the assumption of Section IV that each mission has a unique priority (PO). When multiple missions have the same priority, we must be able to impose a total ordering among them, so that we can determine the "best" subset of missions that can be satisfied. Sec. V.A presents some valid objective functions that may be used to evaluate the 'goodness' of a selected set. As maximizing each of these objectives turns out to be hard, we shall develop and evaluate some heuristics that help establish a strict order among missions in the same priority-equivalence class.

V.A. Defining the objective function for missions of equal priorities

The optimality of a particular selection or ordering among missions with the same PO value may be evaluated using several potential metrics:

1) Total Utility of Missions ($\sum_{\forall m \in M} U_m$): This metric measures the total utility of all the missions (both normal and prioritized), without consideration of whether or not mission demands are satisfied. This conventional metric is inappropriate for us, as it does not penalize the inability to satisfy some mission demands.

2) Total Number of Satisfied Missions (I): This metric measures the ability of the network to meet the minimum demand of missions without regard to their utility. However, this problem of maximizing the number of satisfied missions,MAX-CARD SATISFIED MISSIONS, is NP-hard.

Theorem 1. MAX-CARD SATISFIED MISSIONS *is NP-hard, even to approximate.*

Proof. Given a MAXIMUM INDEPENDENT SET (MIS) instance $I = (V, E)$, we produce a problem instance I'. For each node $v \in V$, create a mission m_v with $D_v = 1$ and a sensor s_v. Define the utility function $U_v(x_v) = x_v$. Set $c_{ks} = 1$ for all k, s. We say that two sensors s_u, s_v conflict with each other (i.e., are in some clique ℓ) iff nodes u, v are adjacent in I. Because all c_{ks} are unit, each value x_s is bound by 1. Given the demands and utility functions, x_s must exactly equal 1 if s is to provide any benefit to a mission. Thus m_u, m_v can have their demands satisfied iff their sensors s_u, s_v do not conflict. Thus the nodes corresponding to a max-size conflict-free set of satisfied missions in I' are a MIS in I. □

3) Total Utility of Satisfied Missions ($\sum_{\forall m \in SF} U_m$): This metric combines the goals of the previous two metrics, measuring the total utility of only the satisfied missions (regardless of the number of satisfied missions). However, this problem, MAX SATISFIED UTILITY, is NP-hard too.

Theorem 2. MAX SATISFIED UTILITY *is NP-hard, even to approximate.*

Proof. This proof follows from the proof for Theorem 1. It can be observed from the proof for Theorem 1 that the utility for any *satisfied* mission equals 1 (recall the upper bounds on x_s). Thus in the produced instance I', maximizing the satisfied utility is exactly the same as maximizing the number of satisfied missions. □

4) Weighted Utility of Satisfied Missions ($I * \sum_{\forall m \in SF} U_m$): This metric measures the total utility of only the satisfied missions and weighs it with the total number of satisfied missions. As we show in [13], this is a mixed-integer (MINLP) optimization problem which is well-known to be NP-hard and even popular techniques such as branch-and-bound fail to provide a good solution.

V.B. Heuristics for Solving Prioritized NUM with Missions of Equal Priority

In this section, we describe four heuristics for ordering the demands of missions with equal priority. We can use these heuristics in one of the protocols discussed in Section IV for breaking the tie among missions of equal priority.

V.B.1. Maximum Utility

A simple heuristic is to order the missions in descending order of their minimally utility demands, thus evaluating missions greedily based on their utility demands. Hence, for demands of the form $f_i(X) > D_i$, we order based on the D_is. From the protocol perspective, we simply use the demand value as a secondary priority, which is used for computing the highest or lowest priority (depending on whether Incremental, Batch or Hybrid protocol is used) at each node, when two missions have the same PO.

V.B.2. Minimum Resource

In this heuristic, we give preference to the mission with lower resource requirement, i.e., the whose demand is met at lower source rate. By conserving the capacity of the network, this heuristic should increase the number of missions with satisfied demands. For a mission with a demand $U(x) > D$, the rate at which the demand is met is given by $U^{-1}(x)|_D$. However, since the utility of a mission depends on different sources and different numbers of them, we normalize this quantity (assuming identical rates from each sensor) to obtain the "Per-source Rate" requirement. For instance, for a demand $log(x) + 2log(y) > D$, the value of this heuristic is $e^{D/3}$. This heuristic has no implementation overhead as, the missions just make their (normalized) $U^{-1}(x)$ values available along with their demands.

V.B.3. Maximum Normalized Utility

This heuristic attempts to balance the dual requirements of high utility and low resource consumption. We order the missions based on their "Utility per-bit", preferring the mission with higher value. For a mission with demand $U(x) > D$, its utility per bit is given as $\frac{D}{U^{-1}(D)}$

V.B.4. Virtual Link

This heuristic also attempts to balance the trade-off between high utility and low resource consumption. Intuitively, this approach uses the concept of 'virtual links' introduced in [11] to first estimate the amount of additional bandwidth that is needed to satisfy *all* demands, and then eliminates those missions that depend maximally on the 'virtual flow' to satisfy their demands. Let us assume an imaginary link from each source to its destinations, with infinite capacity; any demanded utility is thus inherently feasible. However, each mission incurs a penalty for receiving data on the virtual link. Suppose a mission has a utility function $U(x_s) = log(1 + x_s)$ and it has a demand $U(x_s) > D_s$. With the introduction of a virtual link, its demand can now be met if $log(1+x_s+x'_s) > D$, where x'_s is the rate transmitted through the virtual link from source s. The penalty $Y(X')$ for using the virtual link can be any concave function of the virtual rates to the mission. The objective function of the network is now given by:

$$\textbf{maximize} \sum_{m\in M} U_m(X_m) - Y_m(X'_m) \textbf{over } x_s \geq 0 \textbf{ subject to}$$

$$\text{i)} \sum_{\forall(k,s)\in l} \frac{x_s}{c_{k,s}} \leq 1, \text{ii) } f_i(X, X') \geq D_i, \quad \forall i \in \{1, \ldots, H\}$$

We can show that the network converges at the unique maximum if the sensor adjusts both its 'real rate' x_s and its virtual rate x'_s (which it does not actually transmit) such that the gradient is driven to zero. This is further explained in [13]. Each mission computes its dependence on the virtual link for meeting its demand, *i.e.,* it computes $Dep_i(X) = 1 - f_i(X)/D_i$. The mission that depends the most on the virtual link is dropped during the Batch protocol. This virtual process is simulated in parallel, along with the actual Batch rate-change protocol. The simulated rates and clique costs are carried along with the real rates and costs in the data packet; as a part of the mission's state, its Dep_i values are also maintained. When two or more missions are selected to be dropped, the Dep_i value is used to break a tie. Thus we see that while it increases the size of the packets, this heuristic, like the rest, does not incur any communication overhead. However, this heuristic works only with the Batch protocol, because this provides only a partial ordering among missions, that is used to decide which mission has to be eliminated.

	$\sum_{\forall m\in M} U_m$	$\sum_{\forall m\in SF} U_m$	I	$I \sum_{\forall m\in SF} U_m$
Min Resource	0.989	0.805	0.945	0.857
Max Util	0.975	0.957	0.839	0.927
Norm. Util	0.981	0.895	0.881	0.920
Virtual Link	0.994	0.854	0.971	0.928

Table 1: Performance of heuristics w.r.t different optimality metrics

V.C. Evaluation of the Heuristics

In this section we evaluate the performance of the heuristics. First, we evaluate how close each heuristic gets to the optimal, for the different metrics of optimality discussed in Sec. V.A. For this, we simulated a 20-node network of random topology with 5 sources and 10 missions and tested the heuristics in MATLAB. We also computed the optimal values of the metrics using GAMS [12].

Table 1 shows the value of each metric (normalized by the optimal, computed by exhaustive search) obtained by the four heuristics. As expected, the "Minimum Resource" heuristic satisfies more missions, while the "Maximum Utility" heuristic yields higher utility of the satisfied missions. Although "Minimum Resource" increases the number of missions satisfied, it does not increase the weighted (satisfied) utility of the network. In contrast, the Virtual Link heuristic not only increases the number of missions satisfied, but also increases the weighted (satisfied) utility of the network (attains $\approx$ 92% of the optimum). Normalized Utility also results in about the same value of $I * \sum_{\forall m\in SF} U_m$ and it yields lower I and higher $\sum_{\forall m\in SF} U_m$ compared to Virtual Link. We also simulated the heuristics in Qualnet for a set of 10 missions belonging to three priority classes. As expected, there was no significant differences in the convergence times for the heuristics.

VI. Conclusion and Future Work

In this work we have extended the NUM framework and developed practical distributed protocols, to address the scenario where all demands not collectively feasible and the cases where priorities are unique and non-unique. In future, we plan to further study the performance behavior of the heuristics and develope techniques to improve the convergence speed of the protocols.

References

[1] F.Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications, Vol.8 (1997) 33-37.*

[2] F.P.Kelly, A.K.Maulloo, D.K.H.Tan. Rate control for comm. networks: shadow prices, proportional fairness and stability. *JORS Vol.49(1998)*

[3] S.Eswaran, A.Misra, T.La Porta. Utility-Based Adaptation in Mission-oriented WSN. *Proc. of IEEE SECON, June 2008.*

[4] http://www.qualnet.com

[5] C. Liu, L. Shi, B. Liu. Utility-Based Bandwidth Allocation for Triple-Play Services. *Proc. of ECUMN, Feb 2007.*

[6] P.Dharwadkar, H.J.Siegel, E.K.P.Chong. A Heuristic for Dynamic Bandwidth Allocation with Preemption and Degradation for Prioritized Requests. *Distributed Computing Systems, Apr. 2001.*

[7] S.H.Low, D.E.Lapsley. Optimization flow control,I: Basic algorithm and convergence. *IEEE/ACM ToN, Vol.7, 861 - 874.*

[8] D. Saha, S. Roy, S. Bandyopadhyay, T. Ueda, S. Tanaka A dist. feedback control mechanism for priority-based flow rate control for QoS provisioning in ad hoc wireless networks with dir. antenna. *IEEE ICC, June'04.*

[9] D. Palomar and M. Chiang. Alternative dist. algorithms for network utility maximization: Framework and applications. *IEEE TAC, March'08*

[10] P. Hande, S. Zhang, and M. Chiang. Distributed rate allocation for inelastic flows. *IEEE/ACM ToN, Vol.15,No.6,1240-1253*

[11] R. Gallager and S. J. Golestaani. Flow control and routing algorithms for data networks. *Proc. 5th Int. Conf. Computer Comm., 779-784, 1980*

[12] http://www.gams.com

[13] S.Eswaran, M.P.Johnson, A.Misra, T.La Porta. Distributed Utility-Based Rate Adaptation for Prioritized Missions. *TR NAS-TR-0090-2008,Dept. CSE, PSU, May 2008.*

[14] L.Bui, R.Srikant, A.Stolyar. Optimal resource allocation for multicast flows in multihop wireless networks. *Proc. of IEEE CDC, December 07.*