Wireless Networks Revenue Optimization through Overload Control with Priority Services

Haitao Lin, Preetam Ghosh, Prabir Das
Converged Multimedia Networks (CMN) Systems Engineering, Nortel, USA
Email: {haitaolin, preetagh, daszprab}@nortel.com

Abstract— **Wireless networks are currently experiencing more overload situations than their wireline counterparts because of explosive mobile traffic growth, unpredictable traffic behavior, service differentiated traffic shedding, etc. Even though extensive research work on network overload control in general is going on, the economic aspect of the overload problem has received very little attention. Managing the incoming traffic in a way that generates the maximum possible revenue under overload warrants special attention. In this paper, we study a realistic wireless switch overload model where message exchange and message discarding at multiple nodes are considered. We propose a distributed overload control framework considering different service types to obtain the optimal revenue while maintaining the switch's capability to handle call attempts during overload situation. This is achieved by exchanging overload information among the nodes that can have a global view of the switch-wide overload situation and its impact on revenue, and hence can adjust their own traffic shedding to improve the revenue generation. Next, we extend the proposed framework to incorporate different call priorities and discuss the conditions for reaching the optimal revenue that ensures preferential treatment to the high priority service types.**

Index Terms— **Distributed control framework, nonlinear optimization, revenue maximization, Priority Services.**

I. Introduction

Many of the existing overload control mechanisms ([1], [2], [3], [5]) can be applied to or has been tailored for wireless switches. However, because of the specific environment that the wireless network is evolving into (such as wireless services and users behavior), new challenges are becoming more and more evident. Prior works in this area try to ensure the proper working of the wireless network under overload. It has rarely been studied how a wireless network can obtain optimal revenue under overload situations. More specifically, while most of the mechanisms control overload by discarding certain amount of traffic, it has received little attention how this traffic shedding can be done in an optimal way so that the network is not overloaded while the revenue is maximized.

The wireless switch becomes complicated as the traffic load and number of services it supports are growing. In most vendors' implementations, a wireless switch is no longer a single node, but consists of multiple nodes that work together to handle the user traffic and the signalling. A call[1] setup procedure includes several rounds of message exchanges among these nodes, while overload traffic shedding could happen in any of these nodes. However, in most of the existing literature, a call setup process in a node is only modeled as a single message processing. There has been little analysis on the overload performance where the message flows are considered.

In this paper, we address the overload control in the context of a multi-node wireless switch, with the purpose of optimizing the revenue generation while maintaining a fixed bound on the delay at each node. In particular, we propose a distributed overload control mechanism that can be applied to wireless (and wireline) switches.

The rest of the paper is organized as follows: Section II briefly discusses the state-of-the-art in single/multi-class overload control algorithms. Section III outlines a multi-node wireless switch model, upon which our overload control framework is applied. Section IV gives detailed description of the proposed overload control algorithms where we do not consider call priorities. Section V discusses the implementation issues and VI presents the simulation results to show that our proposed framework maintains key performance metrics of a switch under overload while improving the revenue generation. In Section VII, we present an extension of the distributed framework that will support call types having different priorities. Section VIII concludes the paper.

II. Related Work

Both overload and congestion control have been studied extensively for telecommunication networks and the Internet. Several single-class overload control techniques for telecommunication networks have been proposed in [1], [2], [4], [12]–[15], [19]. Two popular overload control techniques for telecommunication networks are:

- Call gapping: here all service requests are rejected during certain time gaps. The period and duration of these gaps are determined based on overload conditions.

This paper is based on "A Distributed Overload Control Framework for Revenue Maximization in Wireless Switches," by H. Lin and P. Das, which appeared in the Proceedings of the IEEE Consumer Communications & Networking Conference (CCNC), Las Vegas, USA, Jan. 2006.

[1] In this paper, a "call" not only refers to the voice calls in wireless network, but also refers to any wireless services, such as short messages, packet data sessions, etc.

- Call percentage blocking: here a certain percentage of service requests are blocked during overload periods, and the remainder serviced. The blocking percentage is determined based on the overload level.

There exist several works on multi-class overload control schemes as well. The overload control algorithms proposed in [7] fall in the class of percentage blocked algorithms, which can be applied to call gapping algorithms as well. A token-based scheme was proposed in [4] where an incoming call of a given class is accepted if there are free tokens of that class or if there are free tokens in a common pool of unused tokens. However, as they do not consider any priority mechanisms in token usage, calls belonging to lower priority classes might exhaust the tokens of the common pool under overload. In [17], different queue size thresholds were used, one for each class, for discarding call messages when the thresholds are exceeded in a tail drop manner. There are also some recent works on local overload control considering distributed denial of service (DDOS) attacks [8].

Overload control in the Internet has been achieved through active queue management of router queues. Most of the proposed multi-class active queue management approaches including RIO [18], WRED [20] and [21] are multi-class extensions of the Random Early Discard (RED) algorithm [22]. However, none of the above works consider distributed overload control where better performance can be achieved by using the call drop scenario at other nodes.

Distributed overload control algorithms in the Internet were proposed in [5], [9]–[11] which concentrated on maximizing the call completion rates. Our paper, in turn, considers the revenue maximization at each node of the wireless switch. The major characteristics of our framework are as follows:

- The proposed overload control framework considers the interactions among multiple nodes that work together to complete the setup of a call. When each node makes overload control decisions, the message interactions with other nodes are taken into account. A multiple switch scenario is considered in [2], [16], where the call setup is modeled as a sequence of setup processes beginning from the switch where the call is originated to the switch where the call is terminated, with each switch considered as a single queue. However, in the wireless domain, the call setup is more complicated within a switch because it always includes coordinating a number of resources, such as radio channel, Base Transceiver Station (BTS), Base Station Controller (BSC), etc. Therefore, the call setup process can consist of more than twenty messages exchanged among different nodes, while every node can have its own overload control mechanism.
- The proposed framework supports multiple types of services. The wireless networks support a multitude of services, while the message interactions to setup all these services are different.
- The current wireless switch can handle pretty high level of overload and the mechanisms always focus on how to maintain good performance such as the predefined network delay and throughput under overload, but rarely considers whether the overload control yields the optimal revenue. Our proposed mechanism provides the overload performance from the economic perspective, i.e., improving the revenue generation. This is achieved by coordinating the message discarding at all the nodes that have overload control capability. When one node is reacting to overload by reducing traffic level, how other nodes are throttling the traffic is also taken into consideration. Through traffic shedding in a distributed way, the switch-wide revenue optimization is obtained.

A preliminary version of this work appeared in [6]. In this longer version, we also enhance the distributed framework to support different service type priorities which is more realistic in a wireless switch. Although, the initial framework can perform overload control by achieving the optimal revenue, the extended framework can achieve the same if the switch scheduler follows a specific queueing model as discussed later on. In other cases we can achieve near-optimal results in terms of revenue maximization. This is however necessary to ensure that the high priority call types are dropped less although they generate lesser revenue.

III. Switch Model and Overload Control Framework

In this section, we present the model of a wireless switch on which our proposed framework is to be applied.

Because of the complexity of setting up calls in a wireless network, the switch needs to coordinate a lot of resources when setting up a call. Our proposed overload control framework can be applied to a switch consisting of any number of nodes. For the purpose of presentation in this paper, we assume a three-node switch, as shown in Fig. 1. These nodes are responsible for different tasks

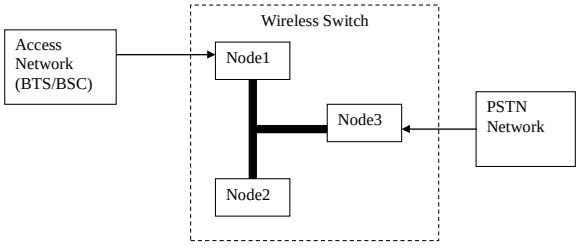


Figure 1. A simplified model of a wireless switch

in the switch, such as: reserving and setting up radio channels for the mobile at the access networks, querying Home Location Register (HLR) or Visitor Location Register (VLR) regarding the user profiles, searching and establishing trunks for a call going out of the switch, etc. A successful call setup includes coordinating all these task, and therefore involves a lot of message exchanges among

these nodes. In Fig. 2, we show a call setup process, which is modeled as a sequence of messages exchanged among all these nodes. The sequence of message exchange is also called a *call flow*. A wireless switch can support many services, while each of the service has a different call flow.

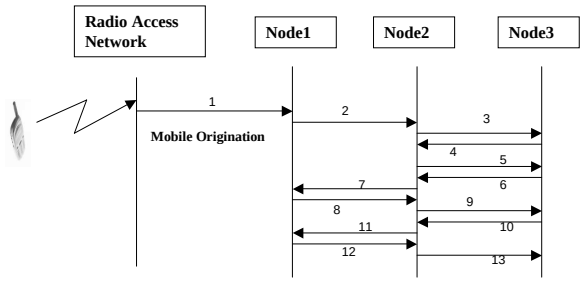


Figure 2. An example of call flow

Our proposed distributed overload control framework consists of two parts: (i) overload control functionalities inside each node; (ii) the overload information exchanges among all the nodes in the switch. The overload functionalities are illustrated in Fig. 3.

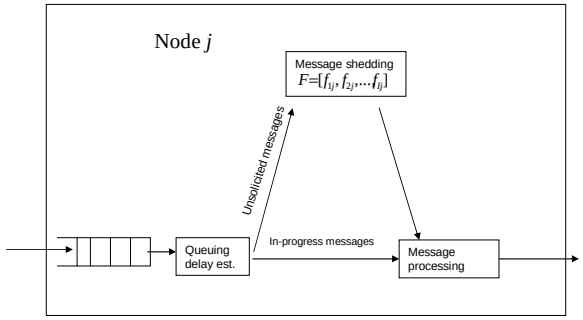


Figure 3. Overload Control Function in a Node

The messages going into a node are as follows:

- Unsolicited messages: the first message in a call flow at each node, e.g., in Fig. 2, message 1 at node1, message 2 at node2, message 3 at node3. These messages initiate the whole call setup process.
- In-progress messages: the rest of the messages. These are mostly responses to a specific message in the call flow. For example, messages 4, 5, ... , 13 in Fig. 2.

Only the unsolicited messages are candidates for discarding when the node is in overload.

When a message is dequeued from the message queue, the message queuing delay is first estimated. If the message is an unsolicited message, the incoming traffic load of this type is updated, and the unsolicited message is sent to the overload control unit, which decides whether this message is discarded or not, depending on the distributed overload control algorithm that will be discussed in detail in the next section. If the message is not discarded, it is sent to the message processing unit. In-progress messages are directly sent to message processing unit. Discarding the first messages in the call flow results in the whole call flow being stopped. On the other hand, discarding the in-progress messages would waste the processing power that has already been spent on this call flow, thus reducing the network throughput.

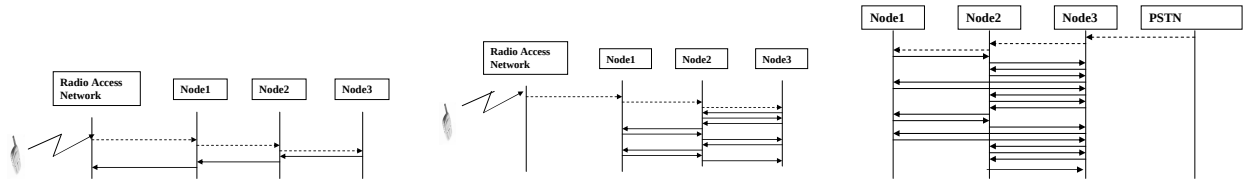
The second part of the overload control framework is the overload information exchange among nodes. As we can see from the switch model, discarding an unsolicited message has global impact on the load to other nodes, as well as impact on the revenue generation (discarding an unsolicited message results in a call failure, thereby losing the potential revenue from the call attempt). In our framework, the message discarding factors, which decide how much traffic of each type is discarded at each node, are exchanged under overload conditions. The details of the information being exchanged will be discussed in Section IV.

IV. Distributed Overload Control Mechanism

We first provide a list of the notations that will be used when presenting the distributed overload control algorithm:

- I: the total number of services the switch supports. For the simplicity of presenting the algorithm, we assume $I = 3$ in this paper.
- N: number of nodes involved in overload control in the switch. As shown in Fig. 1, $N = 3$.
- D_j: the observed average message queuing delay at node j, $1 \leq j \leq N$.
- f_{ij}: message discarding factor at node j for service type i, which is the fraction of calls of service type i allowed into node j, while $(1 - f_{ij})$ of the calls are discarded at node j, where $1 \leq i \leq I$ and $1 \leq j \leq N$.
- T_{ij}: observed call arrival rate of service type i at node j.
- P_i: the per call revenue from a type i call, $1 \leq i \leq I$.
- K_{ij}: in the call flow for service type i, the number of messages going into node j.
- C_j: the message processing time (service time) at node j, if the message is not discarded.
- c_j: the message processing time (service time) at node j, if the message is discarded.
- B_j: the delay bound at node j. For a switch, the delay that is allowed at each node is tightly controlled to ensure the total call setup time is under the QoS limit. It is required that the average delay at node j does not exceed B_j.

Our distributed overload control algorithm takes into consideration the call flow of each service type. Fig. 4 shows all the call flows of the three services assumed in this section. The call flows represent the typical call types supported in a switch: short call flows with minimum message interaction, as in call flow 1, long call flows with extensive message exchange, as in call flow 3, calls generated from the access network, as call flow 2, and calls generated from the network side, as in call flow 3.



(a) Service Type 1 (b) Service Type 2 (c) Service Type 3

Figure 4. Call Flows

From the point of view outside the switch, assuming the offered traffic as $\mathbf{O} = [O_1, O_2, O_3]$, the total revenue is:

$$R = \sum_{i=1}^{I} P_i O_i \prod_{j=1}^{N} f_{ij} \quad (1)$$

The purpose is to optimize the revenue in a distributed environment. Let us take node 1 as an example to describe the algorithm in detail. In node 1, the total revenue of the switch can be estimated as:

$$R_1 = P_1 T_{11} f_{11} f_{12} f_{13} + P_2 T_{21} f_{21} f_{22} f_{23} + P_3 T_{31} f_{31} \quad (2)$$

It is important to point out that at a local node, how the revenue is calculated depends on the specific call flows, as the node calculates revenue based on the traffic observed locally. For example, for service type 1, because the traffic can be discarded at nodes 2 or 3 after it passes through node 1, the total revenue generated from service type 1 should include the message discarding factor at all three node: $P_1 T_{11} f_{11} f_{12} f_{13}$. For service type 3, when node 1 receives the unsolicited message, it is already after the message discarding at node 3 and 2, therefore, only the message discarding factor at node 1 needs to be considered: $P_3 T_{31} f_{31}$. Though the total revenue formulation for three nodes are similar, they are distinct depending on the node and call flows:

$$R_2 = P_1 T_{12} f_{12} f_{13} + P_2 T_{22} f_{22} f_{23} + P_3 T_{32} f_{32} f_{31} \quad (3)$$

$$R_3 = P_1 T_{13} f_{13} + P_2 T_{23} f_{23} + P_3 T_{33} f_{31} f_{32} f_{33} \quad (4)$$

In Equation 2, node 1 only has control over three variables: f_{11}, f_{21}, f_{31}. The adjustment of these three variables has to satisfy two requirements: first, maintain the message delay at node 1 below B_1; second, optimize the revenue generated by the switch. For node 1, the problem is:

Find $0 \leq f_{11} \leq 1$**,** $\leq f_{21} \leq 1$ **and** $0 \leq f_{31} \leq 1$ **such that** R_1 **is maximized, with the constraint:**

$$0 \leq D_1 \leq B_1 \quad (5)$$

In our proposed framework, we assume for each node, the message processing time C_j and c_j are known[2]. Therefore, for each node, the message queueing and processing process can be modeled as a M/G/1 queue:

$$D_1 = \overline{X} + \frac{\lambda \overline{X^2}}{2(1-\rho)} \quad (6)$$

where $\rho = \lambda \overline{X}$, while λ, $\overline{X}$ and $\overline{X^2}$ are the message arrival rate, first and second moment of message processing time respectively. We derive these parameters as follows.

To obtain the first and second moment of the message processing time, we need to find the distribution of the message processing time. There are two message processing times: C_1 and c_1, processing time of messages not discarded and discarded, respectively. Normally $c_1 \ll C_1$. On average, for service type 1, the number of messages per second not discarded at node 1 is given by:

$$M_1 = T_{11} f_{11} f_{12} f_{13} K_{11} + T_{11} f_{11} (1 - f_{12} f_{13})$$

where $T_{11} f_{11} f_{12} f_{13} K_{11}$ accounts for the number of messages of the calls not discarded at any of the nodes, while $T_{11} f_{11} (1 - f_{12} f_{13})$ accounts for the messages of the calls discarded not at node 1, but at either node 2 or 3. Similarly, for service types 2 and 3, the average number of messages per second not discarded at node 1:

$$\begin{aligned} M_2 &= T_{21} f_{21} f_{22} f_{23} K_{21} + T_{21} f_{21} (1 - f_{22} f_{23}) \\ M_3 &= T_{31} f_{31} K_{31} \end{aligned}$$

And the number of messages discarded :

$$\begin{aligned} m_1 &= T_{11}(1 - f_{11}) \\ m_2 &= T_{21}(1 - f_{21}) \\ m_3 &= T_{31}(1 - f_{31}) \end{aligned}$$

Therefore,

$$\lambda = \sum_{i=1}^{I} M_i + \sum_{i=1}^{I} m_i \quad (7)$$

$$\overline{X} = \frac{C_1 \sum_{i=1}^{I} M_i}{\sum_{i=1}^{I} M_i + \sum_{i=1}^{I} m_i} + \frac{c_1 \sum_{i=1}^{I} m_i}{\sum_{i=1}^{I} M_i + \sum_{i=1}^{I} m_i} \quad (8)$$

$$\overline{X^2} = \frac{C_1^2 \sum_{i=1}^{I} M_i}{\sum_{i=1}^{I} M_i + \sum_{i=1}^{I} m_i} + \frac{c_1^2 \sum_{i=1}^{I} m_i}{\sum_{i=1}^{I} M_i + \sum_{i=1}^{I} m_i} \quad (9)$$

Note that, in the expression for $\overline{X^2}$, we consider $variance = 0$, though this can be generalized easily for positive variances. Since the constraint is non-linear, the problem is a non-linear programming problem. However, under overload situation, there is no extra processing power for solving such computationally

[2]To verify capacity, switch vendors conduct extensive characterization on the processing time for their switches, which is called "call timing" or "message processing time". Therefore it is reasonable to assume that the message processing time is a known factor.

expensive problem. Therefore, we reduce the solution space by only allowing 11 discrete values for f_{ij}: $f_{ij} \in \{0, 0.1, 0.2, \ldots, 1\}$. This changes the problem to an integer programming problem, which is still very expensive to find even sub-optimal solutions. This problem can be remedied by moving the computation offline. No details on how this can be done is discussed here, because of the limited space. The basic idea is to generate a mapping between the inputs, such as the observed traffic load of each service and the discarding factors of other nodes, to the outputs, i.e., the optimal discarding factors for the node itself, and store the mapping as a lookup table in the nodes for real time look-up.

Each node broadcasts its message discarding factors (node j broadcasts $f_{ij}, 1 \leq i \leq I$) to other nodes. Upon reception of these factors, each node updates its own message discarding factors. For node j, the updating algorithms is shown in Fig. 5.

```
if D_j > B_j
    f_ij = f_ij − 0.1  for 1 ≤ i ≤ I  until f_ij reaches 0
else
    Find optimal value for f_ij for 1 ≤ i ≤ I by solving the
    integer programming problem outlined in this section.
    Let the solution for f_ij be f'_ij.

    Update the current f_ij as follows:
    f_ij = (f_ij + f'_ij)/2
    then broadcast these values to all other nodes involved
    in the overload control in switch.
endif
```

Figure 5. Message discarding factor update

Basically, if the delay is above the delay bound required, the shedding factors keep decreasing until the observed delay is within the bound. What distinguishes our overload control from others is the way the shedding factors are updated when the observed delay is below the bound. Our algorithm adjusts the local discarding factors based on the global message discarding information (the message discarding information from others). The global information here plays an important role of helping the node accurately estimate two things: the total revenue that can be generated, and the expected delay at the node, with assumed local message discarding factors. The discarding factor adjustment is not done in a single step, i.e., to directly make $f_{ij} = f'_{ij}$. Instead at every step, f_{ij} is adjusted to be the median of the current value and the calculated optimal value, i.e., $f_{ij} = (f_{ij} + f'_{ij})/2$. Our experiments indicated that this approach helps the discarding factors converge, while the former option could cause oscillation in certain circumstances.

V. Implementation Considerations

The following issues can arise while implementing our proposed framework in a real wireless switch:

TABLE I.
Parameters Used in Simulation

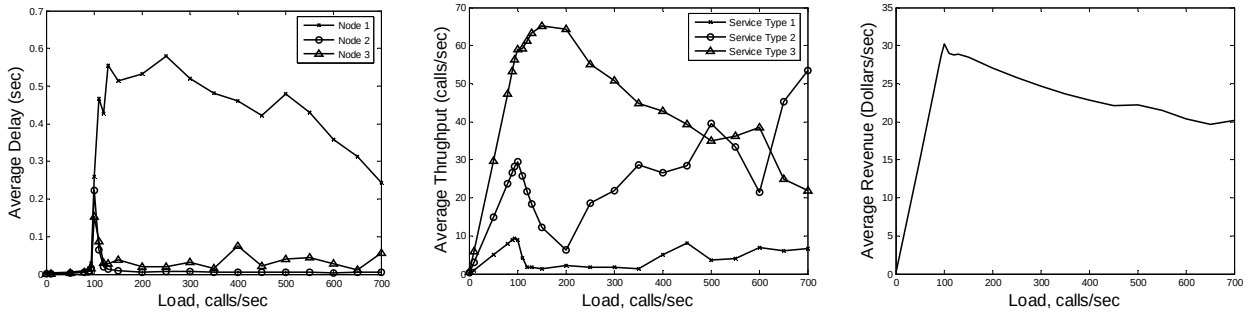
Service Type	1	2	3
Mix	0.1	0.3	0.6
Revenue/call (P_i)	0.1	0.2	0.4
Node #	1	2	3
Delay Bound (B_j)	1	1	1
Msg. Proc. Time (C_j)	0.002857	0.001471	0.001449
Msg. Proc. Time (c_j)	0.0005714	0.0002941	0.0002899

- Message exchange: our distributed overload control requires the message discarding information exchanged among nodes. There are two options: (i) periodical message exchange, i.e., discarding factors are exchanged in fixed time intervals, (ii) dynamic message exchange, i.e., whenever a node updates its own message discarding factors, it broadcasts this information to others. Option 1 has limited overhead, when the interval is not too short, but it may slow down the nodes' reaction to the traffic increase under overload. Option 2 has faster reaction time, and better convergence, but it is more costly to the nodes. In our simulations, we used option 1, with a 1 second interval. We believe message exchange at such frequency would not be a significant cost to the nodes.
- Real time cost of optimization: it is extremely expensive and unrealistic to obtain the optimal solutions (optimal discarding factors) in real time, as it is an integer programming problem. One solution would be to move the computations offline and generate a mapping between the inputs (such as the observed traffic load of each service and the discarding factors of other nodes) to the outputs (i.e., the optimal discarding factors for the node itself). The mapping is stored as a lookup table in the nodes. When a node is executing the algorithm, it only needs to look up the table for the closest input that it can match with the real time parameters, and use the corresponding discarding factors as the solution. Since the solutions are generated offline, we can use expensive algorithms such as genetic algorithm or simulated annealing, to obtain high quality sub-optimal solutions. It is out of the scope of this paper to discuss how these algorithms can be applied.

VI. Simulation Results

To verify the framework, we developed a simulation model, exactly following the switch model in Section III, call flows in Fig. 4 and the proposed overload control framework. The call model, i.e., the mix of all three types of calls in the offered traffic, as well as other important parameters are shown in Table I.

The three nodes are configured to have the same capacity to support 100 calls per second. We run the offered traffic up to 700 calls per second, which is 7 times



(a) Delay (b) Throughput (c) Revenue

Figure 6. Simulation Results

the capacity. Such high level of offered traffic (7× the capacity) covers extreme overload cases.

For an overload control mechanism, the most important performance metric is the delay. Fig. 6(a) shows the average delay at different loads. It is clear in the figure that for all three nodes, the delay is controlled well within the allowable delay bound. On the other hand, we also observe that node 1 has much higher delay than the other two. This is because node 1 is the node accepting two of the three types of calls from outside. Unlike the node inside, such as node 2, even when node 1 discards messages to prevent itself from overloading, the time it spends on the discarded messages can become very large when the traffic load is high, hence the longer delay.

Fig. 6(b) shows the over all average throughput of the three types of calls. It shows differentiation in discarding different types of calls. Call flow of service type 3 has the most number of messages involved, which means it costs the most processing time at the nodes. On the other hand, it also generates the highest per call revenue. Our results show that the algorithm still provide higher throughput for such calls.

Finally, Fig. 6(c) shows the average revenue generated during overload. It is shown that as the offered load increases, after it reaches the switches capacity, the revenue decrease slowly. The decrease is due to the fact that the total processing time every node spends on discarded messages increases as the offered traffic load increases, hence more time wasted on non-revenue generating processing. It can be easily seen that the revenue at the capacity is the maximum revenue a switch can generate. Our results show that even at very high offered traffic load, the revenue is still maintained at close to 66% of the maximum. However, this value is significantly affected by the message processing times C_j and c_j. When discarded messages cost less processing time, we would see higher revenue under overload.

VII. Extending the Framework to Support Different Call Priorities

Note that the proposed framework tries to optimize the total revenue earned at each switch which involves the different call types. However, there might be specific call types that generate less revenue but have higher priority over the other call types e.g., 911 calls etc. Such calls need to be retained by the service providers even though they are not profitable from a revenue perspective. We need a mechanism that can distinguish between such high priority call types and ensure that they are retained in the system.

In this section, we propose an extension of our distributed overload control framework to incorporate call priorities.

A. Overload Control Architecture at a Node

The major design change required to support call priorities is to have *multiple* queues corresponding to the different priorities. The framework discussed above assumes all call types have the same priority and hence could be put in the same queue. Note that each queue, in this extended model, apply to a particular call priority. Multiple call types having the same priority, will enter the same queue. Fig. 7 gives an overview of the new switch model.

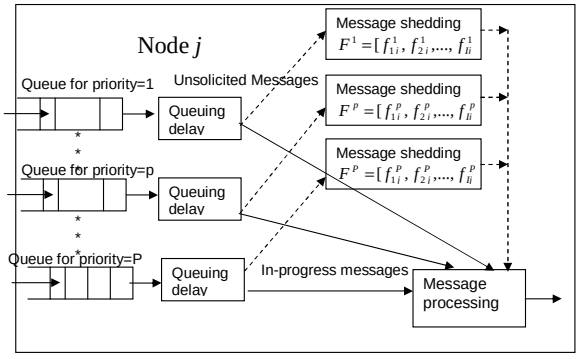


Figure 7. Extended Overload Control Architecture

For each queue, we will have a separate queueing delay estimator function as the expression for queueing delay will be the same for all different call types entering the same queue. Similarly, the unsolicited messages of each queue are passed through its own message shedding function where the message discarding factor for only the call types entering that particular queue are computed. The in-progress messages for all the different call types

(irrespective of priority) are however sent to the Message Processing unit as before.

B. Extended Problem Formulation

In the corresponding problem formulation, we need to incorporate the call priorities. Following are the new variables that need to be introduced:

- P: The total number of call type priorities, and equivalently the total number of queues.
- I: this is now redefined as the total number of call types having the same priority p, $(1 \leq p \leq P)$. Note that we assume that the total number of call types having the same priority (and hence entering the same queue) is the same (for all priorities) for simplicity of notations. However this can be easily generalized to different number of call types having the same priority by making the entries corresponding to the non-existent call-types zero. Hence, the total number of call types (i.e. services) supported by the switch can be as many as $P \times I$ (instead of I as in the previous model).
- D_j^p: the average message processing delay for the calls at queue p, $(1 \leq p \leq P)$, of node j, $(1 \leq j \leq N)$.
- f_{ij}^p: message discarding factor at node j for call type i having priority p, where $1 \leq p \leq P$, $1 \leq i \leq I$ and $1 \leq j \leq N$.
- T_{ij}^p: the observed call arrival rate of call type i having priority p at node j.
- P_i^p: the per call revenue for a type i call with priority p.
- K_{ij}^p: the number of messages going into node j in the call flow for service type i with priority p.
- R_j^p: the total revenue obtained from the calls in the queue corresponding to priority class p at node j.

As an example, let us assume the call types 1, 2 and 3 (as depicted in Fig. 4(a)-4(c)) belong to the same priority class p. Thus following the concept used in Eqns. 2-4, we have:

$$R_1^p = P_1^p T_{11}^p f_{11}^p f_{12}^p f_{13}^p + P_2^p T_{21}^p f_{21}^p f_{22}^p f_{23}^p + P_3^p T_{31}^p f_{31}^p$$
$$R_2^p = P_1^p T_{12}^p f_{12}^p f_{13}^p + P_2^p T_{22}^p f_{22}^p f_{23}^p + P_3^p T_{32}^p f_{32}^p f_{31}^p$$
$$R_3^p = P_1^p T_{13}^p f_{13}^p + P_2^p T_{23}^p f_{23}^p + P_3^p T_{33}^p f_{31}^p f_{32}^p f_{33}^p$$

Similarly, the revenue expressions for the other call types in the different priority queues for the different nodes can be formulated depending on their message flows. Now the total revenue earned at switch j can be expressed by:

$$R_j = \sum_{p=1}^{P} R_j^p \tag{10}$$

C. Optimization problem at node j

Note that the optimization problem formulation also has to change to incorporate the call priorities. Node j has control over $P \times I$ variables corresponding to the different f_{ij}^p's, $(1 \leq i \leq I,\ 1 \leq p \leq P)$. However, if the optimization goal at node j is set as $maximize\{R_1\}$ as before, the high priority calls that generate less revenue will be dropped more. As a result, instead of a global optimization of the revenue generated at node j, the main idea is to *optimize the revenue generated at each queue having priority* p, $(1 \leq p \leq P)$. Hence, the optimization problem needs to calculate only I variables for each queue corresponding to the different f_{ij}^p's, $(1 \leq i \leq I)$ and can be expressed as follows:

For the queue corresponding to priority class p at node j,

$$\text{Find } 0 \leq f_{1j}^p \leq 1, 0 \leq f_{2j}^p \leq 1, ..., 0 \leq f_{Ij}^p \leq 1$$
$$\text{such that } R_j^p \text{ is maximized, with the constraint}$$
$$0 \leq D_j^p \leq B_j$$

Now, this optimization goal can ensure that the high priority call types are not affected by the low priority ones because the non-linear programming problem does not calculate the message discarding factors corresponding to these call types (as they are in a different queue $p' \neq p$). Also, computing the discarding factors for each queue independently, will also guarantee an *optimal solution*, even without a global optimization of R_1. This is because the unknown variables (f_{ij}^p's) involved with each term in $\sum_{p=1}^{P} R_j^p$ are independent. hence we can write:

$$\max_{f_{ij}^p, 1 \leq i \leq I, 1 \leq p \leq P} \{R_j\} = \max_{f_{ij}^1, (1 \leq i \leq I)} \{R_1^1\} + \max_{f_{ij}^2, (1 \leq i \leq I)} \{R_1^2\} + ... + \max_{f_{ij}^P, (1 \leq i \leq I)} \{R_1^P\} \tag{11}$$

However, the low priority (and high revenue generating) call types can still play a role in the determination of the high priority call type discarding factors because they will affect the delay constraint. Next we show how the queueing discipline at the node affect the discarding factors for high priority calls and show the calculations for D_j^p.

D. Estimation of the Processing Delay, D_j^p

Note that we need to incorporate priority queueing disciplines to manage the call types with different priorities. This is to ensure that the high priority calls are in no way affected by the low priority ones. The optimization problem outlined above computes the message discarding factors separately for the call types belonging to different queues. Let us consider two different queues corresponding to priorities p and p', $(p > p')$. Thus the processing delay for calls in queue p should be independent of the arrival rate of calls into queue p', such that the delay constraints in the optimization problem are independent for the different queues. This can only be achieved by implementing a *preemptive priority queueing strategy* in the scheduler. Assuming a preemptive priority M/M/1 queueing model at the scheduler we can write:

$$D_j^p = \frac{\overline{X^p}}{1 - \rho^p} \tag{12}$$

where $\rho^p = \lambda^p \overline{X^p}$, while λ^p and $\overline{X^p}$ are the average message arrival rate and first moment of message processing time respectively for calls entering the p^{th} queue which can be derived similarly as before and concisely presented here:

$$
\begin{aligned}
M_1^p &= T_{11}^p f_{11}^p f_{12}^p f_{13}^p K_{11}^p + T_{11}^p f_{11}^p (1 - f_{12}^p f_{13}^p) \\
M_2^p &= T_{21}^p f_{21}^p f_{22}^p f_{23}^p K_{21}^p + T_{21}^p f_{21}^p (1 - f_{22}^p f_{23}^p) \\
M_3^p &= T_{31}^p f_{31}^p K_{31}^p \\
m_1^p &= T_{11}^p (1 - f_{11}^p) \\
m_2^p &= T_{21}^p (1 - f_{21}^p) \\
m_3^p &= T_{31}^p (1 - f_{31}^p) \\
\lambda^p &= \sum_{i=1}^{I} M_i^p + \sum_{i=1}^{I} m_i^p \\
\overline{X^p} &= \frac{C_1^p \sum_{i=1}^{I} M_i^p}{\sum_{i=1}^{I} M_i^p + \sum_{i=1}^{I} m_i^p} \\
&+ \frac{c_1^p \sum_{i=1}^{I} m_i^p}{\sum_{i=1}^{I} M_i^p + \sum_{i=1}^{I} m_i^p}
\end{aligned}
$$

Note that all the calculations pertain to the p^{th} queue and we assume call types 1, 2 and 3 enter queue p and $I = 3$.

It is important to model the switch using an M/M/1 preemptive priority queueing model instead of an M/G/1 preemptive priority model because the processing delay expression for the M/G/1 counterpart will have a contribution from the low priority call types in the term for residual delay. However, to make the optimization problems for each queue independent we need to make sure that the delay expressions for a particular queue is also not dependent on the call types with lower priority. Similarly, the delay expressions for non-preemptive priority queueing models will have contributions from lower priority call types and hence cannot make the optimization problems at different queues independent. It should be noted, however, that the M/M/1 preemptive queueing model can give us an optimal solution by reaching the global optima that will be reported by our initial model. However, the queueing model that needs to be applied will ideally depend on the switch characteristics, and for cases where this model is not appropriate we can only achieve near-optimal estimates of maximum revenue at each queue, instead of the maximum revenue at the entire switch.

The corresponding algorithm for updating the message discarding factors is quite similar to the one in Fig. 5 and presented in Fig. 8.

Note that the new algorithm has to run for each queue p, $(1 \leq p \leq P)$ at switch j resulting in a run-time complexity of $O(P \times I)$ which is linear in the total number of call types. The algorithm in Fig. 5 also had exactly the same run-time complexity as in the previous case the total number of call types supported was denoted by I instead of $P \times I$ as in the extended algorithm. The message delay at the nodes are hence quite similar to that reported for the initial model. The queueing delay component will however be different due to the M/M/1 priority queueing model at the scheduler.

if $D_j^p > B_j$
$f_{ij}^p = f_{ij}^p - 0.1$ for $1 \leq i \leq I$ until f_{ij}^p reaches 0
else
Find optimal value for f_{ij}^p for $1 \leq i \leq I$ by solving the corresponding integer programming problem.
Let the solution for f_{ij}^p be $f_{ij}^{p/opt}$.
Update the current f_{ij}^p as follows:
$f_{ij}^p = (f_{ij}^p + f_{ij}^{p/opt})/2$
then broadcast these values to all other nodes involved in the overload control in switch.
endif

Figure 8. Extended message discarding factor updation algorithm considering different call priorities

VIII. Conclusions

In this paper, we presented a new framework for overload control, in particular for the multi-node switches. Our framework can support multiple types of services and achieve the maximum revenue at each switch. The overload control mechanism is distributed, but all nodes work together to achieve the improvement on the revenue generated. Our results show that the proposed overload control framework is able to control the message delay at all nodes in the switch, and provides revenue improvement as compared to the normal delay-based overload control mechanism. We also extend the proposed framework to support multiple call priorities that can achieve the maximum revenue if the switch scheduler follows a particular queueing model and near-optimal revenue otherwise.

Acknowledgment

We thank all anonymous referees for their valuable comments and suggestions, which helped significantly improve the quality of the paper.

References

[1] S. Kasera, J. Pinheiro, C. Loader, M. Karaul, A. Hari and T. LaPorta, "Fast and Robust Signaling Overload Control," *the Ninth International Conference on Network Protocols (ICNP'01)*, pp. 323-331.

[2] R. R. Pillai, "A Distributed Overload Control Algorithm for Delay-Bounded Call Setup," *IEEE/ACM Tran on Netw*, 9(6), 2001, pp. 780-789.

[3] S. K. Kasera, R. Ramjee, S. R. Thuel, X. Wang, "Congestion Control Policies for IP-Based CDMA Radio Access Networks," *IEEE Tran. on Mobile Computing*, 4(4), J2005, pp. 349-362.

[4] A.W. Berger and W.Whitt, "The Brownian approximation for rate-control throttles and the G/G/1/C queue," *Journal of Discrete Event Dynamic Systems*, v 2, 1992, pp. 685-717.

[5] W. Park, J. Rhee, J. Won, S. Lee, "An overload control strategy for distributed mobile communication systems," *IEEE Intl. Conf. on Universal Personal Commn. (ICUPC)*, 1998, v 2, pp. 1057-1061.

[6] H. Lin, P. Das, "A Distributed Overload Control Framework for Revenue Maximization in Wireless Switches," *IEEE Consumer Communications and Networking Conference (CCNC)*, 2006, pp. 569-573.

[7] S. Kasera, J. Pinheiro, C. Loader, T. LaPorta, M. Karaul and A. Hari "Robust Multiclass Signaling Overload Control," *Proceedings of the 13th IEEE Intl Conf. on Network Protocols, (ICNP'05)*, Nov 2005.
[8] Y. Kim, W. C. Lau, M. C. Chuah and H. J. Chao "PacketScore: Statistics-based Overload Control against Distributed Denial-of-Service Attacks," *Proc. of 23rd IEEE Conf. on Comp. Commn., (Infocom'04)*, Apr 2004.
[9] S. Saad-Bouzefrane and C. Kaiser "Distributed Overload Control for Real-Time Replicated Database Systems," *Proc. of 5th Intl. Conf. on Enterprise Information Systems, (ICEIS'05)*, pp. 380-388, 2003.
[10] Si Wu and K. Y. Michael Wong "Dynamic Overload Control for Distributed Call Processors Using the Neural-Network Method," *Proceedings of IEEE-NN*, 9(6), 1998.
[11] P. Ayres, H. Sun, H. J. Chao and W. C. Lau "A Distributed Denial-of-Service Defense System Using Leaky-Bucket-Based PacketScore," *Proc. of Appl. Crypt. and Network Security (ACNS '05),* New York, June 2005.
[12] B. L. Cyr, J. S. Kaufman, and P. T. Lee "Load balancing and overload control in a distributed processing telecommunications system," United States Patent No. 4,974,256, 1990.
[13] B. T. Doshi and H. Heffes "Analysis of overload control schemes for a class of distributed switching machines," *Proceedings of ITC-10,* Montreal, June 1983, Section 5.2, paper 2.
[14] M. Rumsewicz "Ensuring robust call throughput and fairness for scp overload controls," *IEEE/ACM Tran. on Netw.,* v 3, pp. 538548, 1995.
[15] B. Wallstrom "A feedback queue with overload control," *Proceedings of ITC-10,* Montreal, June 1983, Section 1.3, paper 4.
[16] A. I. Elwalid and D. Mitra "Analysis, approximations and admission control of a multi-service multiplexing system with priorities," *Proceedings of IEEE Infocom,* Boston, April 1995.
[17] M. Rumsewicz "Analysis of the effects of ss7 message discards schemes for call completion rates during overload," *IEEE/ACM Tran. on Netw.,* v 1, pp. 491-502, 1993.
[18] D. Clark and W. Fang "Explicit allocation for best effort packet delivery service," *IEEE/ACM Tran. on Netw.,* August 1998.
[19] M. Schwarz, "Telecommunication Networks: Protocols, Modeling and Analysis," *Addison-Wesley,* 1998.
[20] Technical Specification from CISCO, "Distributed weighted random early detection."
[21] U. Bodin, O.Schelen and S. Pink, "Load-tolerant differentiaiton with active queue management," *in ACM Comp. Commn. Review,* July, 2000.
[22] S. Floyd and V. Jacobson,"Random early detection gateways for congestion avoidance," *IEEE/ACM Tran on Netw,* 1(4),pp 397-413, 1993.

Haitao Lin received the Ph.D. in Computer Science and Engineering from the University of Texas at Arlington in 2004. He received the B.E. degree in radio engineering from Southeast University, Nanjing, China, and the MS degree in computer applications from the Beijing University of Posts and Telecommunications, Beijing, China. He is currently with Converged Multimedia Networks (CMN) Systems Engineering at Nortel, Richardson, Texas. His research interests include wireless network performance modeling and enhancement, wireless link adaptation, wireless network resource management, and applied game theory. He is a member of IEEE. Email: haitaolin@nortel.com

Preetam Ghosh is a Ph.D. student at the CReWMaN Lab, Univ. of Texas at Arlington. He received his B.E. degree from Jadavpur University, Calcutta, India and MS degree (both in CSE) from Univ. of Texas at Arlington. The work was done when he was a student intern with Nortel, Richardson, Texas. His research interests include Scheduling in Optical networks and QoS issues in Wireless networks.
Email: ghosh@cse.uta.edu

Prabir Das received his Ph.D. in Electrical & Electronics Engineering from University of Strathclyde, Glasgow, UK; B.Sc Eng & M.Sc Eng degrees in Electrical Engineering from BUET, Dhaka, Bangladesh. He is currently the Director of Converged Multimedia Networks Systems Engineering at Nortel, Richardson, Texas. He leads a team of 80 Systems Engineering staff engaged in network performance modeling and characterization of VoIP, IMS/CSCF, CDMA, GSM and UMTS products. His team works with R&D organizations to define and develop high performance multimedia network products and partners with Nortel customers to evolve communication networks with emerging technologies. He has been with Nortel for more than 22 years, holding a number of management positions in Wireline and Wireless network planning, network engineering and systems engineering. He is a Chartered Engineer of IEE and a Member of IEEE.
Email: daszprab@nortel.com