# Effective Switching Scheduling Algorithm Using Concatenated Data Block to Reduce Guard-Time for Opt-Electronic Packet Switch

Takashi Kurimoto, Eiji Oki, Kohei Nakai, Naoaki Yamanaka
NTT Network Service System Laboratories
3-9-11 Musashino shi, Midori cho,
Tokyo 180-8585, Japan

***Abstract*-A new scheduling algorithm that concatenates data blocks to increase switching throughput is proposed. The algorithm controls the degree of concatenation and reduces the number of switching instances to improve switch utilization. The switch architecture uses a virtual output queue switching architecture where the core switch fabric is an optical matrix switch. The optical matrix switch requires the guard-time overhead needed for optical switch control. By reducing the number of switching instances, guard-time overhead can be reduced and switch utilization can be improved. Computer simulations show that efficiency is dramatically increased and that fairness in terms of data throughput among output ports is achieved**

## I. INTRODUCTION

With the increase in traffic on the Internet, core switches located in backbone links must be extended to offer far higher bandwidths. One of best architectures to meet this demand is the virtual output queue architecture [1].

The proposed architecture is based on the virtual output queue with optical switching core, we call this hybrid an opt-electronic switch. The opt-electronic switch architecture combines the advantages of both electric and optical switching. The optical element can transmit huge amounts of data. The current optical switching core is limited by the guard-time needed to offset the slow control speed of optical switches. Because optical element action is slow, of the order of several hundred nano-seconds, the large overhead incurred waiting for the completion of optical switching must be reduced.

This paper introduces the concept of reducing the overhead by concatenating data blocks dynamically. A heuristic algorithm based on this concept is proposed. This algorithm also considers the allowable output bandwidth of each port, scheduling results in fairness of output bandwidth at each port.

Computer simulations are used to evaluate the proposed algorithm and the reduction in switching overhead is confirmed along with output fairness.

The next section describes the overhead incurred by optical switching. The new concept for reducing the overhead is introduced in section 3 and evaluated by computer simulations in section 4. The simulation results show that the new mechanism works well.

## II. Opt-electric PACKET switch

### *A. Opt-electrical packet switch architecture*

Fig. 1 provides a block diagram of the opt-electronic switch architecture [2] adopted in this paper. This switch consists of 4 elements: input line module (ILM), output line module (OLM), scheduler, and optical switch fabric (OSF). ILM, OLM and scheduler are realized as electronic devices, while the OSF is a matrix optical switch consisting of $N$ x $N$ optical switch elements. Each optical switch element offers only on-off action.

The ILMs have a segmentation function, SEG, and virtual output queue, VOQ. Arriving variable length packets are divided into several fixed-size data-blocks in SEG. These blocks are temporary stored in the corresponding VOQ [1]. The scheduler organizes the optical path configuration of the OSF and sets up the connections between input ports and output ports. And data blocks are passed to the OLM via the OSF. In the OLMs, each packet is reassembled from several data-blocks.

Using fixed-size data-blocks simplifies the scheduling algorithm and implementation. However, using small data-blocks increases the switching overhead which lowers switching efficiency.

The next section describes this problem.

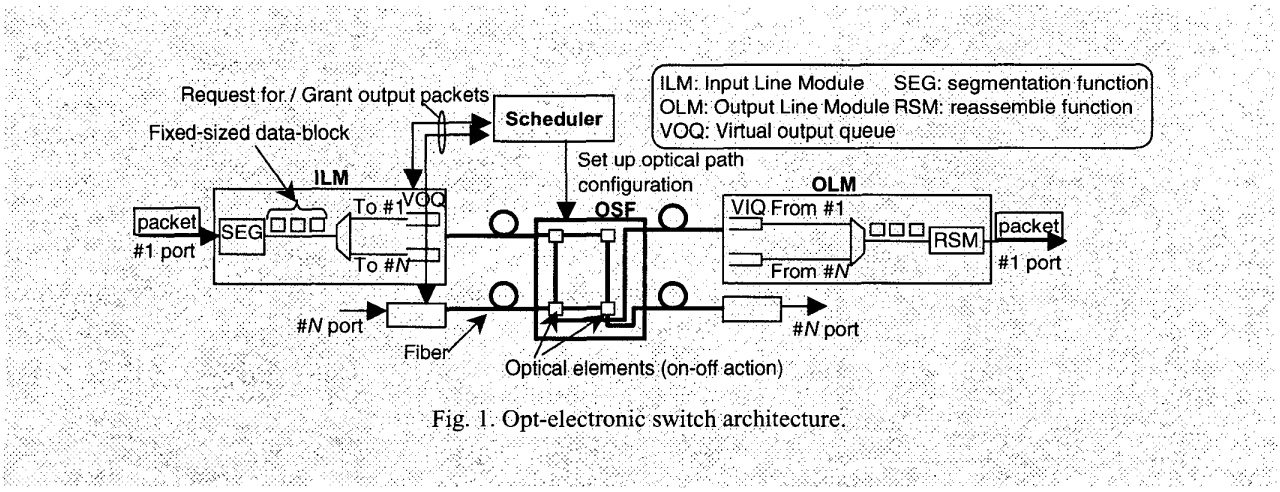



Fig. 1. Opt-electronic switch architecture.

0-7803-7097-1/01/$10.00 ©2001 IEEE

### B. *Guard time overhead to set up optical elements*

Switching overhead is determined by the need to set sufficient guard time ($T_g$) as follows.

Fig. 2 shows the OSF switching process. Arriving packets are segmented into fixed-sized data-blocks and are switched to output ports through the OSF. At $t_0$, $t_1$, $t_2$, the optical paths are re-configured. Note that proper guard time, $T_g$, is required to set an optical path in the OSF. The guard time mainly consists of thee factors: switching action time of optical element, bit synchronization time, and frame synchronization time. The service time of fixed-sized data-block, $T_s$, decreases as data transmission speed increases. If $T_s$ is short, the overhead becomes significant because the guard time is constant.

Here we define efficiency as $T_s /(T_g + T_s)$ and estimate the efficiency for the case in which the data transmission speed of each optical path is 10 Gbps. Fig. 3 shows that the efficiency versus data-block size, for three different $T_g$ values.

This results shows that the overhead cannot be ignored when $T_g$ is relatively long compared to $T_s$ with high-speed links. One way of reducing the overhead is to use long data-blocks, but this fails to support short packet transfer because each data block would be partly empty. This paper presents another approach to reducing the guard time overhead.

## III. New concept based on opt-electric switch

The new concept of dynamically concatenating data-blocks is shown in Fig. 4. Please compare figure 2 to figure 4. In figure 2, three guard times are needed, while only two are needed in figure 4 because the first two concatenated data-blocks use only one guard time. Our concept is that the frequency of guard time usage can be reduced by transferring as many concatenating data-blocks as possible.

In this system, the path configuration is determined for each data-block switching cycle and the data-blocks are concatenated when the path configuration is same as last configuration. To suppress the occurrence of path configuration changes, a new scheduling algorithm is proposed that also offers fairness in loading the output ports.

First, we describe the simple priority-based scheduling algorithm and next we describe the concatenating-weight algorithm based on simple priority-based scheduling.

### (a) simple priority-based scheduling

Fig. 5 shows the simple priority-based scheduling algorithm that achieves fairness among input ports. This figure shows the function block of ILM and the scheduling process.

The ILM consists of a selector, VOQs, priority registers(PRGs) and a quanta distribution function(QDF).

When a packet arrives at an $i^{th}$ ILM from an input line, the selector picks the $VOQ_{i,j}$ corresponding to the packet's destination ($j^{th}$ output port) and stores the packet in the $VOQ_{i,j}$.

To achieve fairness among ports, we use the priority register which is similar to the deficit counter of DRR [3]. A constant value, $Q$, is distributed to all ILMs at every data-block switching cycle. Each QDF calculates the value, $Q_d = Q/N_a$, where $N_a$ is the number of activated VOQs, i.e. those holding packets. After this calculation, the ODF distributes the $Q_d$ value to the PRGs of the activated

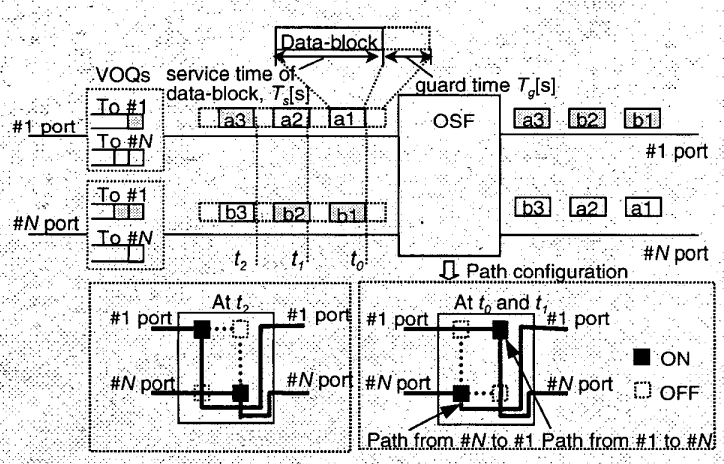



Fig. 2 Fixed size data block switching.

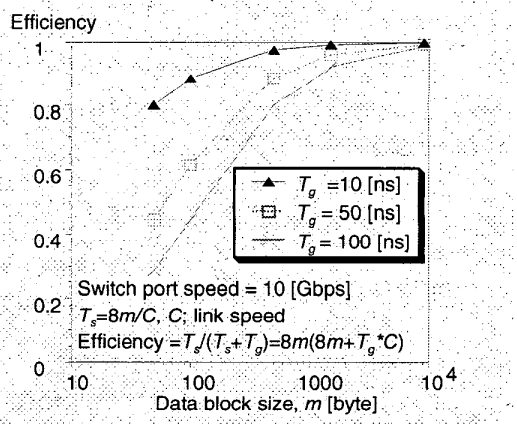



Fig. 3. Impact of overhead guard time on efficiency.

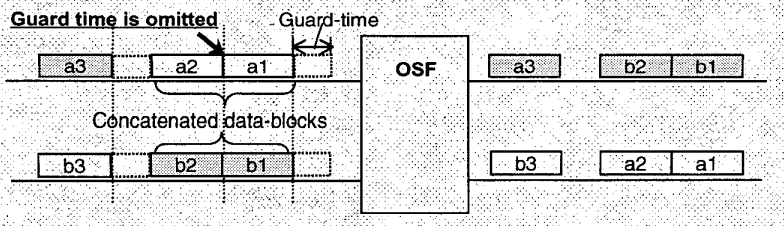



Fig. 4. Proposal of dynamic block concatenation.

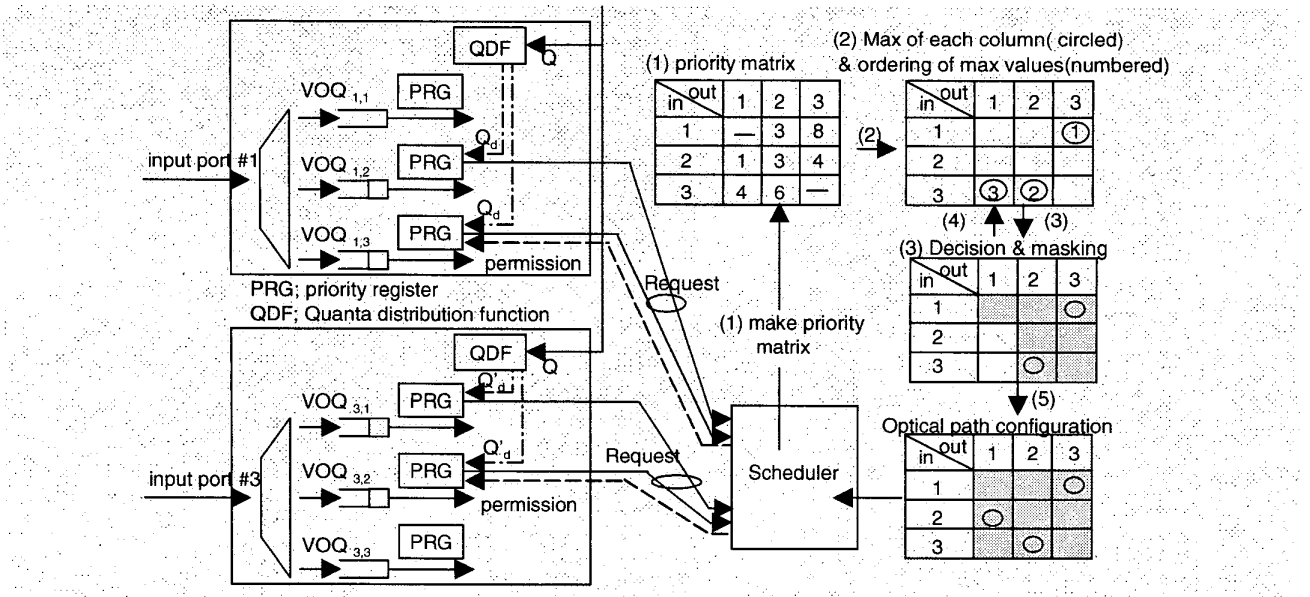



Fig. 5. Simple priority based scheduling algorithm.

buffers. This $Q_d$ value is added to the priority value stored in PRG.

Next, this PRG requests the scheduler to give it permission to output data from its VOQ. These requests carry the priority values stored in the PRGs. Let $P_{i,j}$ be the priority value stored in the $VOQ_{i,j}$, index $i$ indicates the input port number and index $j$ indicates the output port number. With this priority information, the scheduler arranges the optical path configuration as follows.

1st step: The priority matrix shown in figure 5 is made. The column number represents the desired output port while the row number represents the input port number.

2nd step: In each $j^{th}$ column, the maximum number, $P_j^{column} = \max\{1 \le i \le N \mid P_{i,j}\}$ is selected. Note that $N$ is the input/output port number of the OSF. For example, $P_{3,1}$ is maximum among the first column entries. In this figure, the elements of maximum number of each column are represented by circles. Next, these $P_j^{column}$ values are ordered. For example, priority is ordered as $P_3^{column} > P_2^{column} > P_1^{column}$. This order is shown by the numbers within the circles.

3rd step: Decide the connections between input and output ports starting with the highest order priority. For example, the connection between 1st input port and 3rd output port = connection(1, 3) and connection(3, 2) are decided. Connection(3, 1) is blocked because it shares the same output port with connection(3, 2). Next, the matrix is masked at the selected ports and the matrix is redrawn to show the remaining input and output ports. The 2nd step is restarted. When no more connections can be made, this scheduling process is completed.

After this process ends, optical paths are configured and the scheduler give the permission to the corresponding VOQs to output data. At the ILM side, the VOQ that receives permission to output data, subtract the data-block size from the priority value stored in the PRG.

This simple algorithm is mainly intended to achieve fairness. The concatenating-weight algorithm described below reduces the guard-time overhead.

(b) concatenating-weight algorithm

The concatenating-weight algorithm adds a concatenating-weight matrix to simple priority-based scheduling. Fig. 6 shows the process of this new scheduling algorithm. The first step is the same and yields the same priority matrix as simple priority-based scheduling. In the second step, before selecting the maximum row number, the priority matrix is changed so that the same configuration as the current configuration is preferentially determined. The priority matrix is changed as follows.

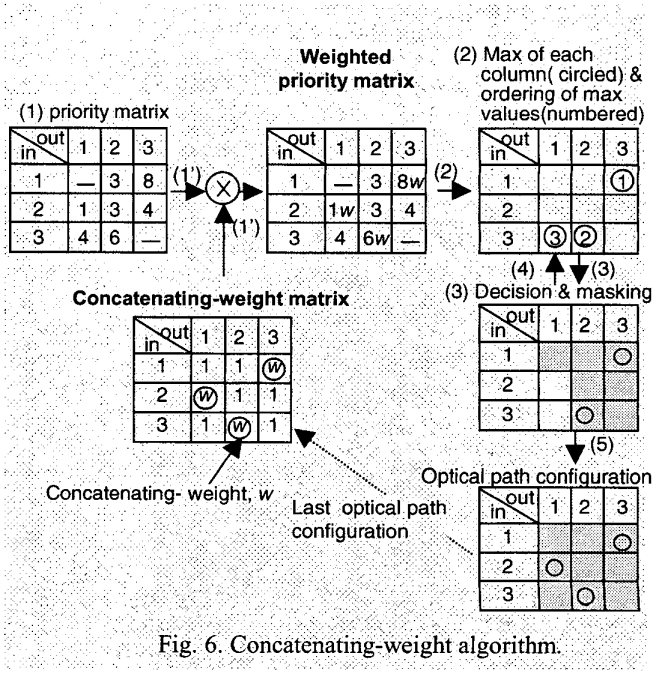



Fig. 6. Concatenating-weight algorithm.

Here we make the concatenating-weight matrix, $C_{i,j}$. The value of $C_{i,j}$ is set to $w$ ($w \geq 1$) when connection$(i, j)$ was connected in the last configuration; The value of $C_{i,j}$ is set to 1 when connection$(i, j)$ was not connected. Next the weighted priority matrix, $P'_{i,j}$, is calculated as $P'_{i,j} = P_{i,j} * C_{i,j}$. This indicates that priority is given to current connections in deciding the scheduling. This minimizes alteration of the optical path configuration. As $w$ increases the probability of more data-blocks being concatenated increases. This minimizes the overhead of the guard time. Parameter $w$ is set by the network operator considering the factors of fairness and overhead.

The remaining steps are the same as in the simple priority-based algorithm.

## IV. Evaluation of PROPOSED method

### A. *gain evaluation*

We used computer simulations to evaluate the performance of the proposed algorithm.

We assumed an $N$ port switch. The data transmission speed via a input port is $C$ [bps]. Variable length packets arrive at the switch and packet length follows an exponential distribution where the average packet length is $L_p$. The average service time of packets, $T_p$, equals $L_p/C$. The interval between packet arrivals at a port follows an exponential distribution and the average interval is $T_i$ [s]. We define the input load, $\rho$, as $T_p/T_i$. The destination ports are randomly decided equally. The service time of each data-block is $T_s$ and guard time is $T_g$. We define values $N_B$ and $N_G$ as follows. During a certain time $T_i$, $N_B$ data-blocks, are switched via OSF and $N_G$ guar-times are inserted. We define the system efficiency as follows.

$$Effcency = N_B * T_s / T_i = N_B * T_s / (N_B * T_s + N_G * T_G)$$

$N_B * T_s$ represents that total service time of data block during $T_i$.
$N_G * T_G$ represents that total required guard-time during $T_i$.
The average concatenating ratio, $R_s$, is defined as follows

$$R_s = N_G / N_B$$

If $R_s$ equal to 1, no data-block is concatenated and no guard time is reduced. While, if $R_s$ equal to 0, all the data-blocks are concatenated and all guard-time is eliminated.

We estimated the efficiency in both switches without concatenating data-blocks and with concatenating data-blocks. Fig. 7 shows the gain by using concatenating data-blocks versus input load for three switch port numbers from 8 to 32. In this simulation, we set parameters $T_g = T_s$ and concatenating-weight parameter $w = 1$. This figure shows that with 8 ports, a gain of about 50% is obtained. As port number increases, the configurations are changed more often and the gain falls.

Fig. 8 shows the fairness in terms of output data amount for each output port versus input load. To estimate the fairness, we use the Fairness Index [4] value. Good fairness is achieved when this fairness index value is 1.

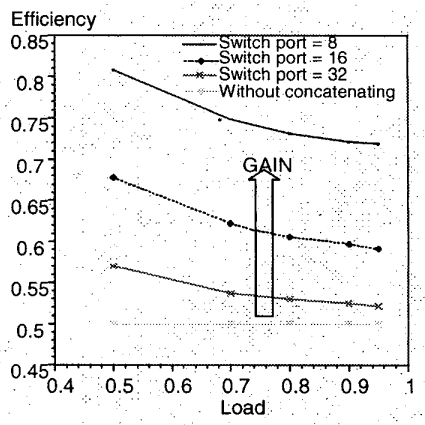
$$\text{Fairness index} = \sum x_i^2 \Big/ n \sum x_i^2$$

$x_i$ = aggregate output data amount from $i^{th}$ port / available output data amount

From this result, we can see that fairness is achieved in the conditions examined.

### B. *concatenating-weight parameter evaluation*

As described in section 4.1, our algorithm offers only slight value when there are many input/output ports. However, it is possible to change its performance by varying the concatenating-weight parameter $w$. Fig. 9 shows the estimated efficiency versus $w$. In this simulation, we set parameters $T_g = T_s$, the input load to 0.98, and port number to 32. The result is that efficiency is nearly 1 when the concatenating-weight parameter is set higher than 1.5. This result indicates that we can use the switching capacity efficiently.




**Simulation condition**
- service time of data block $=T_s$ [s]
- arrival traffic condition
  - average packet service time, $T_p$ [s]
  - $T_p = 5 * T_s$
  - packet service time distribution is exponential
  - output port selection is random
- switch port number $= N$ [port]
- $T_g = T_s$, $w = 1$
- Efficiency(without concatenating) $= T_s/(T_s + T_g)$
- Efficiency(with concatenating) $= T_s/(T_s + R_s * T_g)$

Fig. 7. Gain evaluation

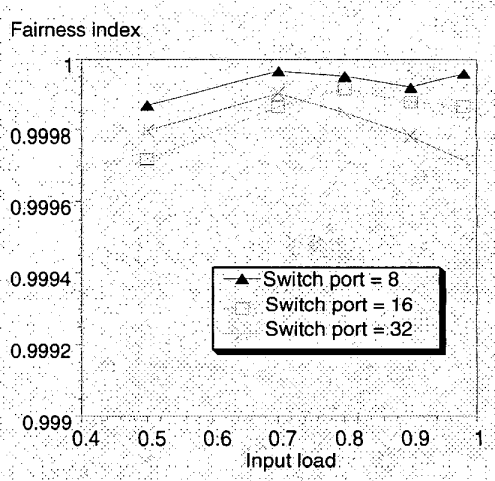



Fig. 8 Fairness among output ports.

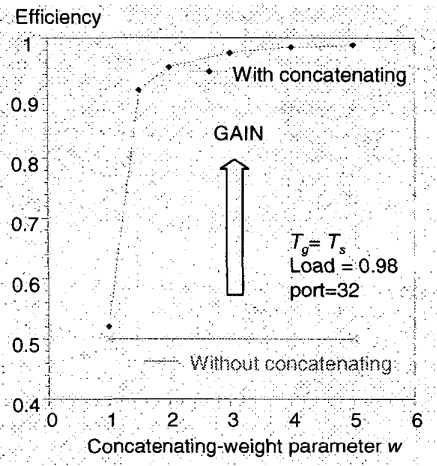



Fig. 9. Gain evaluation.

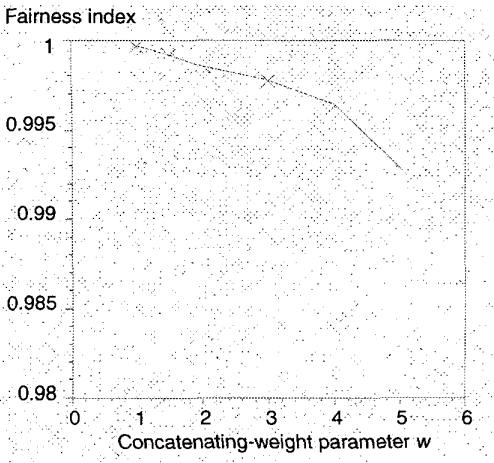



Fig. 10. Fairness among output ports.

Fig. 10 plots the fairness index versus concatenating-weight parameter. This shows that fairness degrades slightly at large parameter values. There is obviously a trade off between switch utilization and fairness. The optimum trade-off must be determined for the installation being considered.

## V. Conclusion

We proposed a mechanism that reduces the occurrence of optical switch rearrangement and improves switch utilization. The proposed mechanism is based on the virtual output queue switching architecture where the core switch fabric is an optical matrix switch. To improve the efficiency of the entire switch, a concatenating-weight algorithm is used. This method suppresses the frequency with the optical switch needs to be rearranged while also achieving good fairness. Large concatenating-weight parameters increase the probability of switching long concatenated data blocks which minimizes the guard time overhead.

## References

[1] Graig Partridge et al., "A 50 Gbps IP Router", IEEE/ACM Transactions of Networking, vol. 6, No. 3, pp.237-247, Jun 1998.

[2] Y. Suemura, S. Takahashi, S. Araki, et al., "Terabit/s Opto-electronic Packet Switch Demonstrator", Technical report of IEICE, SSE99-131.

[3] M. Shreedhar, G. Varghese, "Efficient fair queueing using Deficit Round-Robin", IEEE/ACM trans. on networking, vol. 4, no. 3, June 1996.

[4] Bonomi Flavio, W. Fendick Kerry, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", IEEE Network, vol. 9, no. 2, Mar. 1995, pp. 25-39

0-7803-7097-1/01/$10.00 ©2001 IEEE