An Efficient Multicast Protocol for WDM Star-Coupler Networks

Shiann-Tsong Sheu and Chien-Pang Huang

Department of Electrical Engineering, Tamkang University
Tamsui, Taipei Hsien, Taiwan 30043, R.O.C.
E-mail:stsheu@ee.tku.edu.tw

Abstract

In a wavelength division multiplexing (WDM) star coupler network, both data and signals issued by the transmitters of nodes are combined by the PSC and then broadcasted back to all nodes. To serve a multicast request on WDM star coupler networks, a transmitter of source node, a receiver of each destination node and a data channel have to be available at the same time. The multicast scheduling algorithm proposed in [1], while incorporating arbitrary transceiver tuning times and propagation delays, can not fully utilize the channel bandwidth. In this paper, an efficient backtrack multicast scheduling algorithm (BMSA) for multicast packet transmissions on WDM-based local lightwave networks with a passive star topology is proposed. The simulation results show that the algorithm substantially improves the channel utilization.

1. Introduction

Among various single-hop WDM networks, passive star coupler (PSC) networks are studied most extensively due to their physical simplicity [1], [2], [3], [4]. A generic single-hop star network, as shown in Fig. 1, composed of a passive star coupler and a set of nodes which equipped with a set of transmitters and receivers. In Fig. 1, there are N nodes connected to a single broadcast star coupler. Each node has a fixed-transceiver operates on the dedicated control channel, and m tunable transmitters (TTs) and n tunable receivers (TRs) ($m \geq 1$, $n \geq 1$) are used to tune to proper channels for packet transmissions or receptions. Such a network model is termed as the $CC\text{-}TT^m\text{-}TR^n$ single-hop system [5].

The basic concept of multicast scheduling algorithm (MSA) for WDM star-coupler networks in [1] is briefly described as follows. Assume the tuning time for transmitters and receivers are T_t and T_r, respectively, and the propagation delay caused by the transmission between any two nodes is R. Moreover, the MSA also assumes that the length of a data packet is fixed P bits. At first, MSA determines the earliest time (earliest_rec_time) at which the multicast packet can be received simultaneously by all destination nodes. To obtain earliest_rec_time, MSA finds the timing of earliest free receiver of each node in the multicast group and chooses the latest release time of these founded receivers plus T_r as the earliest_rec_time. Secondly, MSA finds the earliest free time (source_free) of all transmitters belonging to the source node. Then, MSA finds the earliest free channel (trans_chan_time) after source_free+T_t. According to these information, the MSA determines the selected channel to service the packet at earliest_tran_time = max(trans_chan_free, source_free, earliest_rec_time-R). Finally, MSA reserves the time period (earliest_tran_time, earliest_tran_time + P) for this request by updating the free time of the corresponding transmitter, receivers and the assigned channel. After the reservation is done, MSA schedules the next incoming request that is buffered in queue. Since MSA only records the information of the free time of transmitters, receivers and channels, some available channel resources could be wasted.

In this paper, we propose an efficient backtrack multicast scheduling algorithm (BMSA) to fully utilize all resources (i.e., the transmitters, receivers and channels).

2.The Backtrack Multicsat Scheduling Algorithm (BMSA)

For achieving the goal of determining the schedule in a distributed fashion, a dedicated control channel in network is enough. For using this protocol, the transmission channels (include the control channel) are divided into fix-length time slots. The time slots on the control channel λ_0 is further divided into N mini slots, as shown in Fig. 2. Each mini slot contains three fields:

the source node *src*, the destination nodes *dests* and the transmission length (in slots) *len* of the multicast request. The *src* field encodes the source address in $\log_2 N$ bits to minimize the overhead. The *N*-bit *dests* field, one bit for each node, identifies the members belong to the multicast group. If the *j*th bit is set in the *dests* field, node *j* must receive this packet. Before transmission, each node should send a request (with *src*, *dests* and *len*) by using a mini slot to inform all nodes to schedule its transmission time. For a time slot in channel λ_O, only the *i*th node can use the *i*th mini slot.

Consider the example shown in Fig. 3, a new incoming multicast packet with 2 time slots transmission time is issued from source node 1 to destination nodes 2 and 3. Let $req(s,D)$ denote the multicast request issued from node *s* to nodes belong to multicast group *D*, i.e., $D=\{d_1, d_2, ..., d_z\}$. In order to formally present the scheduling algorithm, let $T(s,i) = \{t_1, t_2, ..., t_u\}$ denote a set of unused fragments in the *i*th transmitter of node *s*, in which $t_k = (t\_begin_k, t\_end_k)$ stands for an idle time interval from time t_begin_k to time t_end_k in this transmitter. Similarly, let $R(d,j) = \{r_1, r_2, ..., r_v\}$ denote a set of unused fragments in the *j*th receiver of node *d*, in which $r_k = (r\_begin_k, r\_end_k)$ stands for an idle time interval from time r_begin_k to time r_end_k in this receiver. We also let $C(l) = \{c_1, c_2, ..., c_w\}$ denote a set of unused fragments in the *l*th channel in network, in which $c_k = (c\_begin_k, c\_end_k)$ stands for an idle time interval from time c_begin_k to time c_end_k in this channel.

Fig. 3 shows the case of usage condition of all resources when $N=3$, $m=2$, $n=2$ and $c=2$. We note that $C(1)=\{(2,3),(5,\infty)\}$ and $C(2)=\{(0,0),(4,\infty)\}$ in this example. This implies that channel λ_1 has an unused fragment from time slot 2 to time slot 3, and channel λ_2 has an idle time period at time slot 0. Before describing the way of finding a proper request to use an idle fragment, we define a equation to calculate the intersected idle interval of a pair of intervals as follows. The time interval *a*, denoted as $a=(a\_begin, a\_end)$, is obtained by $F(x\_begin_i, x\_end_i, y\_begin_j, y\_end_j)$ according to the relationships of four input parameters $x\_begin_i, x\_end_i, y\_begin_j, y\_end_j$. That is, $a = (a\_begin, a\_end) = (max(x\_begin_i, y\_begin_j), min(x\_end_i, y\_end_j))$ only if one of the following conditions is satisfied : $(x\_begin_i \le y\_begin_j$ and $x\_end_i \ge y\_end_j)$, $(x\_begin_i \le y\_begin_j$ and $x\_end_i \le y\_end_j)$, $(x\_begin_i \ge y\_begin_j$ and $x\_end_i \le y\_end_j)$ and $(x\_begin_i \ge y\_begin_j$ and $x\_end_i \ge y\_end_j)$. Otherwise, $a=\varnothing$ (empty set).

Now we introduce the backtrack scheduling algorithm. The way of BMSA to find out a suitable time interval for the considered request is following four steps : (1) According to two sets *T*s and *R*s, we employ the function mentioned before to calculate the set of intersected idle intervals of each tunable transmitter of source node and each tunable receiver of destination nodes in the multicast group. Let $TR(s,i,d,j)$ denote the set of all idle intervals which exist in both the *i*th transmitter of source node *s* and the *j*th receiver of destination node *d* at the same time. Consider Fig. 3 for example, the $TR(1,2,2,1)$ only contains the time interval $(5,\infty)$ due to the intersected interval between $T(1,2) = \{(0,0),(4,\infty)\}$ and $R(2,1) = \{(2,3),(5,\infty)\}$ is $(5,\infty)$. (We note that $F(0,0,2,3) = \varnothing$, $F(4,\infty,2,3) = \varnothing$ and $F(4,\infty,5,\infty) = (5,\infty)$.) (2) After all *TRs* are calculated, the BMSA determines the available time intervals for each pair of source node and each destination node. Let $TRSC(s,d) = \{trsc_1, trsc_2, ..., trsc_u\}$ denote the set of all idle time intervals for single connection of source node *s* and destination node *d* $(d \in D)$, where $trsc_k = (trsc\_begin_k, trsc\_end_k)$ stands for an idle time interval from time $trsc\_begin_k$ to time $trsc\_end_k$ in this receiver. (3) If there exists at least one idle time interval in each $TRSC(s,d)$, $d \in D$, then the set of available time intervals for the multicast packet, denote as *TRMP*, is calculated in this step. Each member belongs to *TRMP* are candidates for serving this request. Consider the example shown in Fig. 3 again, the $TRMP(1,(2,3))$ contains two time intervals $(2,3)$ and $(4,\infty)$ which are derived from $TRSC(1,2)$ and $TRSC(1,3)$. We note that the time interval $(0,0)$ is omitted because that its length is not enough to service the mutlicast request $req(1,(2,3))$. In fact, this checking operation can be performed in step (1), (2) or (3). (4) Comparing the obtained *TRMP* with *C*s, the available time intervals in each channel for the multicast packet are easily determined. Let $CMP(l)=\{cmp_1, cmp_2, ..., cmp_u\}$ denote a set of available fragment for the multicast packet in the *l*th channel in network, in which $cmp_k = (cmp\_begin_k, cmp\_end_k)$ stands for a continuous time period from time cmp_begin_k to time c_end_k in this channel. Any time interval in $CMP(l)$, $l=1,2, ...,c$, is a candidate for this multicast request. For illustration, consider the example shown in Fig. 3 again. The $CMP(1)=\{(2,3),(5,\infty)\}$ and $CMP(2)=\{(4,\infty)\}$ can be reserved for the mutlicast request $req(1,(2,3))$. It is obvious that the time interval $(2,3)$ in $CMP(1)$ is the best choice to assign for this request. That is, the BMSA employs the *best fit* approach to assign the channel to minimize the fragment length and obtain a high channel utilization.

The backtrack scheduling algorithm is now given as follows.

Algorithm BMSA;

Input: An incoming multicast request $req(s,D)$ with transmission time P, and sets $T(s,i)$, $i=1,.2, \ldots, m$, $R(d,j)$, $\forall d \in D$, $j=1,2, \ldots, n$, and $C(l)$, $l=1,2, \ldots, c$.

Output: The consecutive time interval to service request $req(s,D)$.

```
BEGIN
    FOR d:=1 TO #(D) DO
    /* #(D) is the number of nodes in multicast group D */
    BEGIN
        FOR i:=1 TO m DO
        /* m is the number of TTs in each node*/
        BEGIN
            FOR j:=1 TO n DO
            /* n is the number of TRs in each node*/
            BEGIN
                Derive the idle time intervals of
                TR(s,i,d,j) from T(s,i) and R(d,j);
            END;
        END;
        Calculate the idle time intervals of TRSC(s,d)
        from TR(s,i,d,j);
    END;
    Find the time intervals of TRMP(s,D) from TRSC(s,d),
    d∈D;
    FOR l:=1 TO c DO
    BEGIN
        Calculate the available time intervals of CMP(l)
        from TRMP(s,D) and C(l);
    END;
    Find the best fit time interval from CMP(1), CMP(2),
    ..., CMP(c);
END.
```

3.Simulation

3.1.Simulation Models

Four simulation models are investigated in this section. For simplicity, we assume that all nodes connect to the PSC at the same distance and the propagation delay between node and the PSC is ignored. In these simulation models, the packet arrival rate of node i is a Poisson distribution with a mean σ_i, and the packet length is an exponential distribution with a mean of L segments. The node load *(NL)* for node i can be defined as

$$NL_i = \sigma_i \times L.$$

Thus, the network load (denoted as Λ) can be defined as

$$\Lambda = \sum_{i=1}^{N} \sigma_i \times L.$$

In these simulations, the average message length is 5 segments (L=5), and all nodes have the same node load NL ($NL_i = \Lambda/N$, $i=1,2, \ldots, N$) where Λ=15.0. We also assume that the distributions of multicast size and destination nodes for all incoming packet are uniform. Suppose the number of reserved slots for packets on channel i within the observing window is S_i, the average channel utilization ACU is calculated by the following equation:

$$ACU = \frac{\sum_{i=1}^{c} S_i}{c \times W}.$$

For each packet arrives the buffer, its buffer delay of each packet in the scheduling algorithm is measured by counting the number of time slots passed by before its packet is successfully transmitted. In the first simulation, we investigate how the channel utilization and the average buffer delay of packets of the MSA and BMSA are affected by applying different numbers of available channels (c). The other assumptions are listed as follows: (1) N = 5 nodes in the network, (2) m = 2 tunable transmitters in each node, (3) n = 2 tunable receivers in each node, (4) c = {1, 2, 3, 4, 5} data channels on the network.

In the second simulation model, the channel utilization obtained by the MSA and BMSA are measured and compared when the incoming multicast packets with different multicast sizes (N) . The other assumptions are given as follows: (1) N = {2, 3, 4, 5, 6} nodes in the network, (2) m = 2 tunable transmitters in each node, (3) n = 2 tunable receivers in each node, (4) c = 3 data channels on the network.

In the third simulation model, we investigate the channel utilization obtained by the MSA and BMSA under the cases of the numbers of tunable transmitters (m) varies from 1 to 4. The corresponding assumptions are given as follows: (1) N = 5 nodes in the network, (2) m = {1, 2, 3, 4} tunable transmitters in each node, (3) n = 2 tunable receivers in each node, (4) c = 3 data channels on the network.

In the last simulation model, we also observe the channel utilization obtained by the MSA and BMSA under the cases of the numbers of tunable receivers (n) varies from 1 to 4. The other assumptions are given as follows: (1) N = 5 nodes in the network, (2) m = 2 tunable transmitters in each node, (3) n = {1, 2, 3, 4} tunable receivers in each node, c = 3 data channels on the network.

3.2.Simulation Results

Fig. 4 shows the average channel utilization and the average buffer delay of each packet obtained by the BMSA and MSA under different numbers of channels when N=5, m=2, and n=2. Fig. 4(a) illustrates the channel utilization obtained by the BMSA is higher than that of the MSA under different numbers of channels. If network load (Λ) is greater than or equal to the number of channels (c), we say that the network load is heavy loaded. We found that given three channels, the obtained average channel utilization of BMSA is almost equal to 1.0. This implies that all channels in network are fully utilized when the network traffic load is heavy. Fig. 4(b) illustrates the average buffer delay of each packet produced by the MSA and BMSA under different numbers of channels. We can find that the average buffer delay of the BMSA is also better than that of the MSA. We note that the BMSA may assign an unused time interval, which is resulted from scheduling earlier packets, for a latter incoming packet.

Fig. 5 shows how the channel utilization obtained by the BMSA are affected under different multicast sizes of packets in the network when c=3, m=2, n=2 and Λ=15.0. In Fig. 5, we can see that the channel utilization for N=2 is 0.75 only. This is because that all incoming packets are unicast packets. We also note that the channel utilization of BMSA and MSA are equal when the network without multicast (N=2). However, when the multicast size is greater than or equal to 3 ($N\geq3$), the obtained channel utilization of BMSA is still better than that of MSA. (The *ACU* of BMSA is about 0.98 for cases of $N\geq3$.)

The channel utilization obtained by the BMSA and MSA under different numbers of tunable transmitters and different numbers of tunable receivers are illustrated and reported in Fig. 6 and Fig. 7, respectively. Fig. 6 shows the obtained channel utilization for c=3, n=2 and Λ=15.0. Intuitively, node equips with more tunable transmitters will obtain a higher channel utilization. In the worse case (m=1), the BMSA still offers a channel utilization of 0.97. We also note that the channel utilization of MSA is 0.87 even when each node has 4 tunable transmitters. Fig. 7 illustrates the obtained channel utilization for c=3, m=2 and Λ=15.0. Similarly, a higher channel utilization will be obtained when node equips with more tunable receivers. Consider the bottom (top) curve in Fig. 7, we can see that for n=1, the channel utilization is about 82% (47%). This shows that in the worse case (n=1), the BMSA obtains an excellent result. This remarkable improvement is apparently caused by the concept of backtracking and reusing the unused fragment. We can easily see that, in MSA, more tunable receivers for each node will obtain a higher channel utilization. Nevertheless, the maximum channel utilization of MSA is 0.94 only. On the other hand, the maximum channel utilization (≈0.99) of BMSA is obtained when the number of tunable receivers for each node is 2 (n=2).

From these simulation results, we can say that it is not necessary to install a network with too many channels, tunable transmitters and tunable receivers for each node. This makes a possibility to build an inexpensive WDM star-coupler network.

4.Conclusions

In this paper, an efficient backtrack multicast scheduling algorithm (BMSA) for multicast packet transmissions on WDM-based local lightwave networks with a passive star topology is introduced. Simulation results show that for a limited number of channels, tunable transmitters and tunable receivers, the proposed algorithm substantially improves the channel utilization and the average buffer delay of each multicast packet.

5. References

[1] Borella and B. Mukherjee, "A Reservation-Based Multicasting Protocol for WDM Local Lightwave Networks", *Proceedings of IEEE ICC'95*, pages 1277-1281, Seattle, June 1995.

[2] A. Ganz and Y. Gao, "A Time-Wavelength Assignment Algorithm for a WDM Star Network", *Proceedings of IEEE INFOCOM'92*, pages 2144-2150, Florence, Italy, April 1992.

[3] A. Ganz and Y. Gao, "Traffic Scheduling in Multiple WDM Star Systems", Proceedings of IEEE ICC'92, pages 1468-1472, Chicago, July 1992.

[4] G.N. Rouskas and M.H. Ammar, "Multi-Destinaton Communication over Single-Hop Lightwave WDM Networks", *Proceedings of IEEE INFOCOM'94*, pages 1520-1527, Toronto, June 1994.

[5] B. Mukherjee, "WDM-Based Local Lightwave Networks Part I : Single-Hop Systems", *IEEE Networks*, pages 12-27, May 1992.

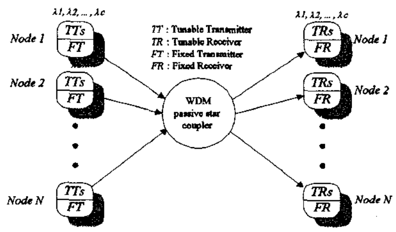


Figure 1.Generic single-hop WDM network with a passive star coupler.

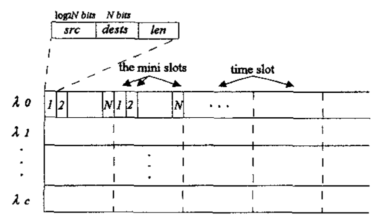
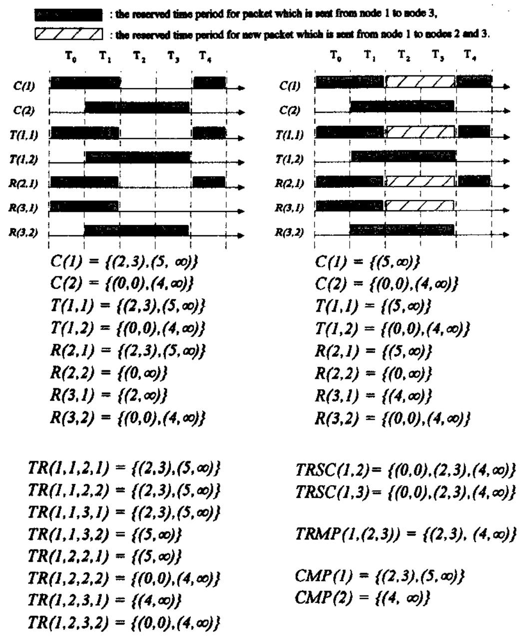
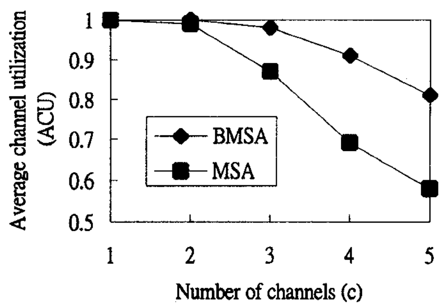


Figure 2. Mini slots in control channel.

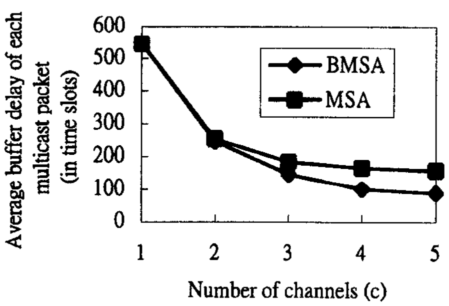


(a) before assignment req(1,(2,3)) (b) after assignment req(1,(2,3))

Figure 3. An example of scheduling a multicast packet *req(1,(2,3))* by BMSA.



(a)



(b)

Figure 4. Comparison of average channel utilization (*ACU*) and average buffer delay of each packet obtained by the BMSA and MSA under different numbers of channels (*c*).

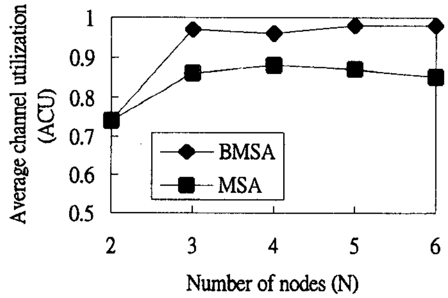


Figure 5. Comparison of average channel utilization (*ACU*) obtained by the BMSA and MSA under different multicsat sizes of packets (*N*).

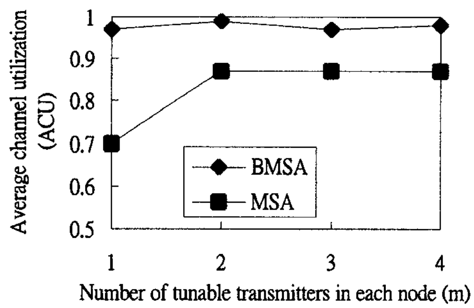


Figure 6. Comparison of average channel utilization (*ACU*) obtained by the BMSA and MSA under different numbers of tunable transmitters in each node (*m*).

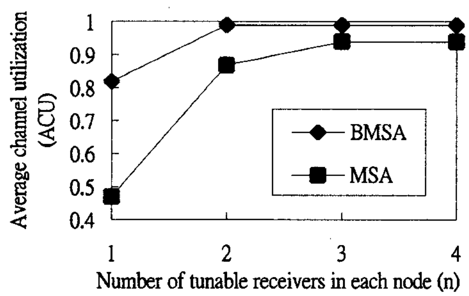


Figure 7. Comparison of average channel utilization (*ACU*) obtained by the BMSA and MSA under different numbers of tunable receivers in each node (*n*).