# Luận văn

## Mô phỏng các giải thuật xếp lịch trên các liên kết đầu ra của mạng OBS

# MỤC LỤC

## CÁC CHỮ VIẾT TẮT

| | |
|---|---|
| AC | Access Control |
| ACK | Acknowledged |
| ASR | Adjustable Synchronous Reservation |
| ARP | Acknowledged reservation period |
| AST | Acknowledged sending Time |
| BAU | Burst assembly Unit |
| BBM | Buffered Burst Multiplexer |
| BFUC | Best Fit Unscheduled Channel |
| BHC | Burst Header Cell |
| BHP | Burst Header Packet |
| CP | Control packet |
| DCS | Data Channel Scheduling |
| DIR | Destination Initiated Reservation |
| DR | Delay Reservation |
| DTWR | Dynamic Two Way Reservation |
| EDFA | Erbium Dopted Fiber Amplifier |
| FDL | Fiber Delay line |
| FFUC | First Fit Unscheduled Channel |
| JIT | Just In Time |
| JET | Just Enough Time |
| INI | Intermediate Node Initiated |
| LAUC | Lastest Available Channel |
| LAUC-VF | LAUC with void Filling |
| NS | Network Simulation |
| NSFNET | National Science Foundation Network |
| NAK | Not Acknowledged |
| NACK | Negative Acknowledged |
| OBS | Optical Burst Switching |

| | |
|---|---|
| OCS | Optical Circuit Switching |
| O/E/O | Optical/Electronic/Optical |
| OPS | Optical Packet Switching |
| QoS | Quality of Service |
| OXC | Optical Cross Connect |
| RWA | Routing Wavelength Assignment |
| SCU | Switch Control Unit |
| SIR | Source Initiated Reservation |
| SOA | Semiconductor optical Amplifier |
| SDH | Synchronous Digital Hierarchy |
| SONET | Synchronous Optical Network |
| SSR | Strict synchronous reservation |
| TAG | Tell and Go |
| TAW | Tell and Wait |
| VF | Void Filling |
| WADM | Wavelength Add-Drop Multiplexer |
| WC | Wavelength Conversion |
| WDM | Wavelength Division Multiplexing |

# MỞ ĐẦU

Trong giai đoạn hiện nay kỹ thuật ghép kênh phân chia theo bước sóng WDM là một giải pháp được lựa chọn để cung cấp một cơ sở hạ tầng mạng nhanh hơn nhằm đáp ứng sự phát triển bùng nổ của Internet. Tuy nhiên, với sự phát triển nhanh chóng của lưu lượng dữ liệu trên mạng, tốc độ xử lý điện tử có thể không còn phù hợp trong tương lai nữa, đồng thời dữ liệu quang thường bị chậm lại do xử lý điện tử tại các node, do đó việc tìm kiếm một phương pháp chuyển tải các gói IP trực tiếp trên lớp quang mà không cần qua chuyển đổi O/E/O cho mạng thông tin thế hệ sau (NGN) là một tất yếu. Nhằm để xây dựng một mạng toàn quang tại đó dữ liệu được duy trì trong miền quang ở tất cả các node trung gian, cần phải thiết kế các giao thức mới dành cho các hệ thống chuyển mạch quang. Một trong các vấn đề cần thiết là làm thế nào để hỗ trợ việc cung cấp tài nguyên nhanh chóng, truyền dẫn đồng bộ (của các gói kích thước biến đổi như các gói IP) cũng như hỗ trợ mức độ cao việc chia sẻ tài nguyên theo thống kê để xử lý hiệu quả lưu lượng có tính bùng nổ mà không cần có đệm ở lớp WDM (do chưa có các bộ nhớ truy cập ngẫu nhiên RAM). Do đó các phương pháp chuyển tải toàn quang cần phải tránh đệm quang càng nhiều càng tốt.

Một vấn đề khác là làm thế nào hỗ trợ chất lượng dịch vụ (QoS) trong mạng Internet quang thế hệ sau. Mạng IP ban đầu cung cấp các các dịch vụ best-effort, tuy nhiên hiện nay các ứng dụng thời gian thực (ví dụ điện thoại và hội nghị truyền hình qua Internet) yêu cầu QoS cao hơn các ứng dụng không phải thời gian thực (như Email hay trình duyệt Web thông thường) và do vậy vấn đề đặt ra đối với lớp WDM là làm thế nào hỗ trợ QoS cho Internet quang. Một số công nghệ khác nhau đang được phát triển, như định tuyến bước sóng (chuyển mạch kênh quang OCS), chuyển mạch gói quang OPS và chuyển mạch chùm quang OBS. Các mạng quang định tuyến bước sóng đã được triển khai và đạt được một số hiệu quả nhất định tuy nhiên các mạng quang định tuyến bước sóng lại lại sử dụng chuyển mạch kênh có thể không phải là công nghệ thích hợp nhất cho các ứng dụng khác nhau sử dụng Internet quang. Kỹ thuật chuyển mạch gói quang là một giải pháp công nghệ

khác và có lẽ là tối ưu hơn cho các ứng dụng mới. Tuy nhiên trong điều kiện một số công nghệ hiện đại như bộ đệm quang, logic quang vẫn chưa thực hiện được thì chuyển mạch gói quang vẫn chưa thể áp dụng vào thực tế. Chuyển mạch chùm quang là công nghệ trung gian giữa chuyển mạch kênh quang và chuyển mạch gói quang đáp ứng được yêu cầu vận chuyển một lượng lớn dữ liệu qua mạng với tốc độ cao và cung cấp các tính năng mới trong giai đoạn tới.

Các vấn đề cần nghiên cứu trong OBS là các giao thức dự trữ và giải phóng tài nguyên, phương pháp thiết lập burst, các giải thuật xếp lịch trên các liên kết đầu ra của mạng OBS. Nội dung đồ án này trình bày tổng quan về mạng OBS trong đó đi sâu tìm hiểu và mô phỏng các giải thuật xếp lịch và quá trình thiết lập burst, mục đích để tìm ra được thuật toán tối ưu nhất cho lượng dữ liệu truyền qua mạng cao nhất và kích thuớc burst cho xác suất mất burst nhỏ nhất để nâng cao chất lượng của mạng OBS.

Nội dung đồ án gồm 5 chương:

- Chương 1: Tổng quan về chuyển mạch chùm quang.
- Chương 2: Kiến trúc mạng chuyển mạch chùm quang
- Chương 3: Báo hiệu và giải quyết xung đột trong mạng OBS
- Chương 4: Các giải thuật xếp lịch trong mạng OBS
- Chương 5:Mô phỏng và kết quả

Phương pháp nghiên cứu của đồ án là mô phỏng các giải thuật xếp lịch trên các liên kết đầu ra của mạng OBS, so sánh kết quả của các giải thuật để từ đó tìm ra giải thuật tối ưu. Ngoài ra đồ án còn nêu lên kết quả mô phỏng quá trình thiết lập burst với 2 trường hợp một mức ngưỡng không có mức ưu tiên, một mức ngưỡng và có một mức ưu tiên để từ đó tìm ra kích thước burst tối ưu cho xác suất mất burst nhỏ nhất.

Trong quá trình  làm đồ án mặc đã cố gắng nhiều nhưng không thể trách khỏi những sai sót, mong các thầy cô thông cảm và hướng dẫn cho em. Để hoàn thành đồ án này em đã được sự hướng dẫn tận tình của thầy Nguyễn Duy Nhật Viễn, em xin gởi lời cảm ơn chân thành đến thầy. Em xin cảm ơn các thầy cô trong

khoa điện tử viễn thông đã truyền đạt cho em kiến thức trong năm năm qua, gia đình, bạn bè em đã hỗ trợ em trong suốt quá trình làm đồ án.

Cuối cùng em xin tỏ lòng biết ơn bố mẹ đã luôn động viên, giúp đỡ và tạo điều kiện tốt để em có thể học hành đến ngày hôm nay.

Đà Nẵng tháng 6/2008

Sinh viên thực hiện

Võ Thị Kim Tuyến

# Chương 1

# TỔNG QUAN VỀ CHUYỂN MẠCH CHÙM QUANG

## 1.1 Giới thiệu chương

Nhu cầu thông tin của con người ngày càng phát triển mạnh mẽ với nhiều loại hình dịch vụ đa dạng. Điều này đặt ra những thách thức đối với hệ thống truyền thông vốn có, vốn được xây dựng chủ yếu phục vụ cho nhu cầu thoại và truyền thông tin không đòi hỏi tốc độ cao.

Một yêu cầu đặt ra là phải xây dựng một hệ thống có khả năng cung cấp băng thông lớn, truyền được một lượng lớn dữ liệu với tốc độ cao. Sợi quang với những tính chất ưu việt cùng việc ứng dụng ghép kênh phân chia theo bước sóng (WDM) là một giải pháp hứa hẹn cho mạng Internet thế hệ mới.

Một mạng toàn quang là mục tiêu hướng tới nhưng trong tương lai gần chúng ta có thể xây dựng một mạng quang trong suốt ít nhất đối với dữ liệu trong đó dữ liệu được chuyển hoàn toàn trong miền quang còn gói tin điều khiển được chuyển trong miền điện. Các công nghệ chuyển mạch quang được đề xuất như chuyển mạch kênh quang, chuyển mạch gói quang và chuyển mạch chùm quang, mỗi công nghệ có các ưu và nhược điểm riêng trong đó chuyển mạch chùm quang dung hòa được những ưu và nhược điểm của hai loại chuyển mạch kia và là công nghệ hứa hẹn trong tương lai.

Nội dung trong chương này là những nét chính về chuyển mạch chùm quang, ưu điểm của nó so với các công nghệ chuyển mạch khác, các phương pháp thiết lập burst trong mạng chuyển mạch chùm quang OBS.

## 1.2 Các thế hệ mạng quang

Thế hệ đầu tiên là kiến trúc mạng point to point WDM (WDM điểm-điểm). Một mạng như vậy gồm nhiều liên kết điểm điểm, ở đó tất cả các lưu lượng đi vào một node từ một sợi quang được chuyển đổi từ quang sang điện và tất cả các lưu lượng đi ra một node được chuyển đổi từ điện sang quang trước khi đưa vào sợi quang. Việc tách ghép luồng quang bằng cách chuyển đổi quang điện tại mỗi node có thể làm tăng độ trễ và tăng chi phí mạng, do đó, để giảm được độ trễ và giảm đi

chi phí mạng ta nên xây dựng một mạng toàn quang nghĩa là việc chuyển tiếp gói hoàn toàn trong miền quang.

Kiến trúc mạng quang thứ hai dựa trên các bộ xen rớt ghép kênh theo bước sóng Wavelength Add-Drop Multiplexer (WADM), trong đó việc tách ghép lưu lượng được thực hiện tại nơi có WADM . WADM có thể tách ra một bước sóng được chọn và cho phép các bước sóng đi qua. Nói chung, lưu lượng đi qua một node thì nhiều hơn lưu lượng cần rẽ tại một node. Do đó bằng việc sử dụng WADM chúng ta có thể giảm được chi phí toàn mạng bằng cách chỉ tách những bước sóng mà đích đến của nó là tại node này còn tất cả các bước sóng khác đi đến node tiếp theo.

Kiến trúc mạng quang thế hệ thứ ba dựa trên việc kết nối các thiết bị toàn quang. Những thiết bị này thường được phân loại thành passive star, passive router và active switch. Tín hiệu được đưa vào một bước sóng tại ngõ vào sao đó công suất tín hiệu này sẽ được chia đều cho tất cả các ngõ ra (sử dụng cùng bước sóng). Một passive router có thể định tuyến một cách riêng rẽ một trong số nhiều bước sóng ở sợi quang ngõ vào đến một bước sóng giống như vậy ở ngõ ra. Active switch cho phép sử dụng lại bước sóng và có thể hỗ trợ những kết nối liên tục qua nó. Passive star được sử dụng để xây dựng một mạng WDM nôi bộ. Trong khi active switch dùng để xây dựng mạng diện rộng định tuyến bước sóng, Passive router dùng như là một thiết bị mux và demux.

## 1.3 Các công nghệ chuyển mạch quang

Hiện tại có 3 công nghệ chuyển mạch quang là chuyển mạch kênh quang Optical Circuit Switching (OCS), chuyển mạch gói quang Optical Packet Switching (OPS) và chuyển mạch chùm quang Optical Burst Switching (OBS). Mỗi loại có đặc điểm riêng và OBS được cho là công nghệ trung gian ở giữa 2 loại kia vì nó dung hòa được ưu và nhược điểm của cả hai và trở thành công nghệ đầy hấp dẫn và hứa hẹn trong tương lai.

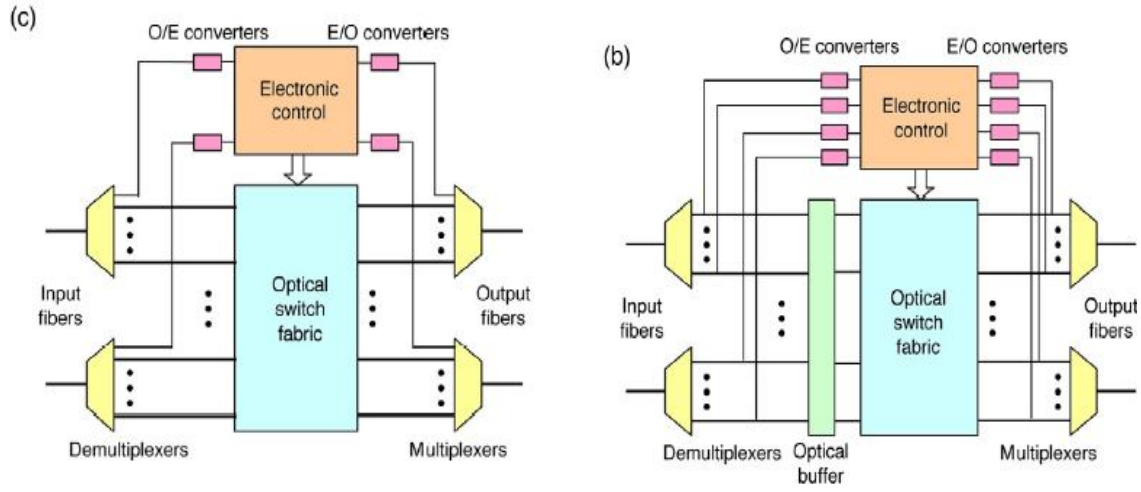
### 1.3.1 Chuyển mạch kênh quang OCS

Chuyển mạch kênh quang hay còn gọi là giao thức định tuyến bước sóng quang Wavelength Routed Networking (WRN) trong đó một đường dẫn quang được thiết lập giữa đích và nguồn trước khi truyền dữ liệu. Trong khi truyền dữ liệu không cần node trung gian thực hiện những công việc phức tạp như xử lý header hay đệm tải trọng. Một đường dẫn quang (light path) được sử dụng để cung cấp một kết nối trong mạng WDM định tuyến bước sóng và có thể trải dài trên nhiều liên kết sợi quang. Các bộ chuyển đổi bước sóng tạo ra các bước sóng khác nhau trên các liên kết quang. Trong mạng WRN băng thông được cấp phát tĩnh hay cố định nên không thể thích ứng với lưu lượng dồn dập và thay đổi cao của Internet một cách hiệu quả. Với một số bước sóng giới hạn cho trước chỉ một số lượng đường dẫn quang hạn chế được thiết lập tại cùng một thời điểm. Nếu lưu lượng thay đổi động, lưu lượng truyền qua các đường dẫn tĩnh sẽ làm cho sự tận dụng băng thông kém hiệu quả. Để có thể đáp ứng được yêu cầu về băng thông lớn trong mạng đô thị và mạng diện rộng, những phương thức truyền tải phải hỗ trợ việc dự trữ tài nguyên và có khả năng truyền được lưu lượng đột biến. Nhưng nếu ta cố gắng thiết lập các đường dẫn quang một cách thức động, thông tin trạng thái của mạng sẽ thay đổi liên tục gây khó khăn trong việc cập nhật trạng thái của mạng. Hơn nữa, dự trữ trong WRN là dự trữ hai chiều trong đó khi có nhu cầu nguồn gửi yêu cầu thiết lập đường dẫn quang và nhận về một xác nhận từ đích tương ứng là kết nối đã được thiết lập cho dù kết nối này có dung lượng bao nhiêu, do vậy việc sử dụng băng thông không hiệu quả về mặt kinh tế.

### 1.3.2 Chuyển mạch gói quang OPS

Chuyển mạch gói quang có thể cung cấp băng thông động nên thích hợp với lưu lượng thay đổi của internet vì nó cho phép chia sẻ thống kê các bước sóng thuộc về các đích và nguồn khác nhau. Trong mạng chuyển mạch gói OPS phần header của mỗi gói được tách ra và xử lý trong miền điện còn dữ liệu phải đệm trong miền quang để chờ header được xử lý xong mới được truyền đi. Vì vậy yêu cầu phải có bộ đệm quang nhưng đây là công nghệ vẫn chưa thực hiện được. Hơn

nữa việc xử lý header trong miền quang không thể thực hiện được trong tương lai gần do chưa có logic quang hoàn toàn nên mặc dù OPS là một công nghệ có nhiều tính năng vượt trội như tốc độ chuyển mạch cao, thích hợp với bản chất của lưu lượng internet nhưng không thực tế trong tương lai gần.




Chuyển mạch burst quang Chuyển mạch gói quang

Hình 1.1 Cấu trúc của OPS và OBS

### 1.3.3 Chuyển mạch chùm quang OBS

Chuyển mạch chùm quang cũng dựa trên ý tưởng tách gói tin điều khiển như OPS nhưng giữa gói tin điều khiển (BHP) và burst dữ liệu có sự gắn kết chặt chẽ về thời gian hơn trong OPS. Các gói tin được tích hợp thành các burst có chiều dài khác nhau và được gửi đi sau gói tin điều khiển một thời gian offset. Thời gian offset được tính toán sao cho gói tin điều khiển được xử lý xong và hoàn thành việc dự trữ tài nguyên tại các node trung gian. Vì vậy công nghệ bộ đệm quang không bắt buộc. Việc xử lý một BHP cho nhiều gói tin cùng một lúc làm giảm thời gian xử lý header cho từng gói trong OPS.

Khác với OCS, OBS sử dụng phương thức dự trữ tài nguyên một chiều truyền dẫn tức thời, nghĩa là burst dữ liệu theo sau một gói tin điều khiển mà không cần chờ chấp thuận của node kế tiếp trên đường đi đến đích nên chiếm dụng tài nguyên hiệu quả hơn OCS. Nó cũng tỏ ra thích hợp với lưu lượng thay đổi đột biến

của internet và theo các kết quả nghiên cứu cho thấy lưu lượng của internet nhất là các trang web có bản chất burst[ 4].

Do có sự thay đổi về độ dài burst mà mạng OBS được coi là ở giữa mạng OPS và WRN. Khi các burst có chiều dài rất nhỏ, gần với các gói thông tin quang thì mạng OBS được coi là mạng OPS nhưng khi các burst có chiều dài khá lớn thì nó có thể coi là mạng WRN. Hơn nữa chuyển mạch chùm quang được thiết kế để khắc phục các nhược điểm của OCS và OPS. Nếu OCS chỉ thích hợp với các dịch vụ tốc độ cố định như thoại hay truyền hình và chiếm dụng tài nguyên lớn, OPS thì tốc độ cao nhưng đòi hỏi các công nghệ chưa thực hiện được như bộ đệm quang hay logic quang thì OBS lại đáp ứng được yêu cầu tốc độ thay đổi của các dịch vụ truyền số liệu và do burst dữ liệu được truyền đi sau các gói tin điều khiển một thời gian offset nên không bắt buộc có bộ đệm quang. Vì vậy OBS được xem như công nghệ chuyển mạch quang hứa hẹn nhất trong tương lai cho một lượng lớn dữ liệu với tốc độ cao.

| Chuyển mạch quang | Sử dụng băng thông | Độ trễ | Tốc độ chuyển mạch | Đồng bộ overhead | Khả năng đáp ứng lưu lượng | Vấn đề chính |
|---|---|---|---|---|---|---|
| OCS | Thấp | cao | Chậm (ms) | Thấp | Thấp | Không linh động |
| OPS | Cao | Thấp | Nhanh (ns) | Cao | Cao | Cần bộ đệm quang |
| OBS | Cao | Thấp | Vừa (có thể ms hay µs) | Thấp | Cao | |

Bảng 1.1: So sánh các công nghệ chuyển mạch

## 1.4. Nguyên tắc thiết lập burst

Thiết lập burst là quá trình tập hợp và đóng gói ở ngõ vào từ lớp cao hơn thành burst tại node biên ngõ vào của mạng OBS. Có nhiều kỹ thuật được đề xuất

trong đó hai kỹ thuật được quan tâm nhất là thiết lập dựa vào bộ định thời (timer-based) và dựa trên mức ngưỡng ( threshold –based).

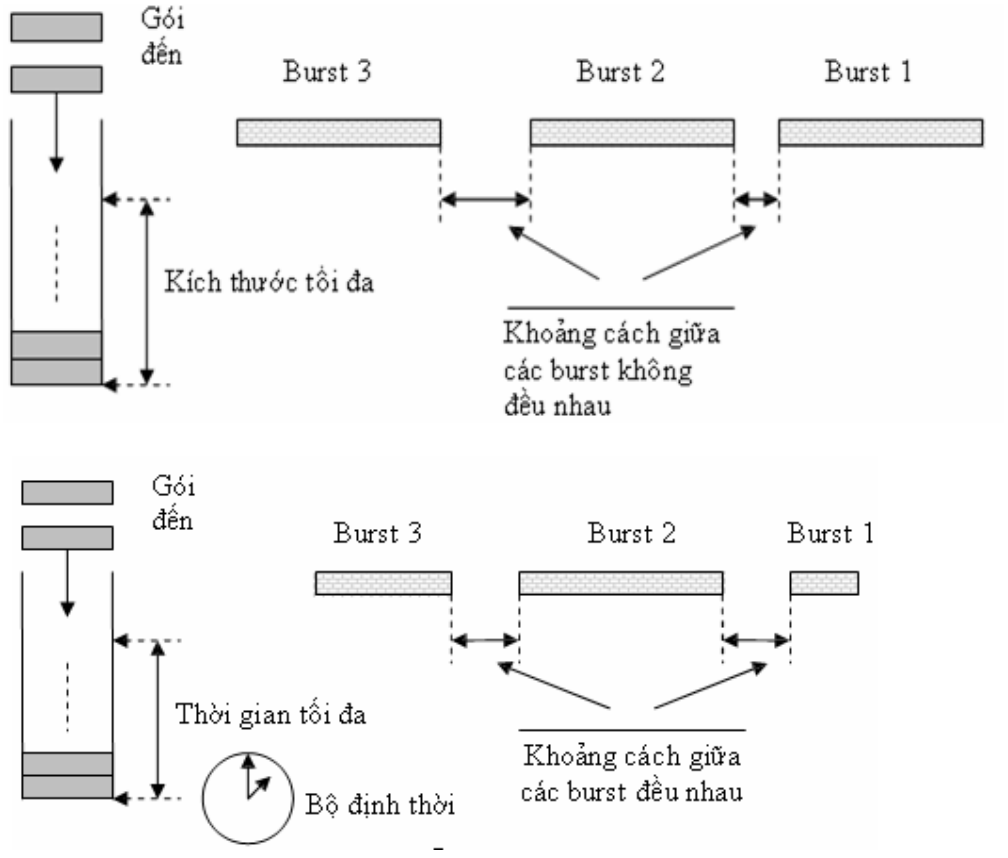
Trong phương pháp thiết lập dựa trên bộ định thời, một burst được tạo ra trong mạng theo chu kỳ thời gian, tức là đúng thời gian đã được định sẵn trong bộ định thời thì sẽ tạo ra một burst không quan tâm đến kích thước burst dài hay ngắn. Do đó, chiều dài của burst biến đổi khi tải vào mạng biến đổi. Trong phương pháp dựa trên mức ngưỡng, số lượng gói trong mỗi burst bị giới hạn hay nói cách khác là chiều dài các burst bằng nhau. Phương pháp đóng gói dựa trên mức ngưỡng sẽ không phát các burst theo một chu kỳ thời gian nào cả. Phương pháp đóng gói dựa trên bộ định thời và dựa trên mức ngưỡng tương tự nhau, bởi vì tại tốc độ cố định cho trước thì về giá trị thời gian hay giá trị kích thước có thể thay đổi qua lại (mapping).

Một vấn đề đặt ra cho thiết lập burst là làm sao tìm ra giá trị của bộ định thời và kích thước ngưỡng để tối thiểu xác suất mất gói trong mạng OBS. Việc lựa chọn một con số tối ưu cho mức ngưỡng (hay giá trị của bộ định thời) là một vấn đề cần nghiên cứu.

Nếu như giá trị ngưỡng quá nhỏ, burst sẽ ngắn, số lượng burst trong mạng sẽ nhiều. Nhiều burst trong mạng dẫn đến nhiều xung đột xảy ra, nhưng số lượng mất gói trung bình trong mỗi lần lại nhỏ. Nhưng với số lượng burst nhiều như vậy sẽ tăng áp lực lên mặt phẳng điều khiển để xử lý các gói điều khiển của mỗi burst dữ liệu. Nếu như thời gian chuyển mạch không được bỏ qua, burst ngắn sẽ dẫn đến việc sử dụng lại tài nguyên trở nên kém đi do phải cần nhiều thời gian cho chuyển mạch. Mặt khác nếu mức ngưỡng quá lớn, burst sẽ dài, số lượng burst vào mạng sẽ nhỏ nhưng số lượng trung bình các gói bị mất trong một xung đột lại lớn hơn nhiều. Do vậy cần một sự cân nhắc giữa số lượng xung đột và số gói mất trong mỗi lần xung đột. Ta cần tính toán để các burst được thiết lập với một kích thước tối ưu để hạn chế đến mức thấp nhất sự mất burst. Tương tự đối với kỹ thuật dựa trên bộ định thời ta phải chọn ra thời gian tốt nhất để kết thúc việc thiết lập burst.

Trong trường hợp các gói chịu sự hạn chế về QoS, như sự bắt buộc có trễ, giải pháp rõ ràng là  thiết lập burst theo thời gian. Giá trị định thời được lựa chọn dựa trên yêu cầu trễ end to end của các gói. Còn trong trường hợp không bắt buộc có trễ, sự thiết lập burst theo chiều dài tỏ ra hợp lý hơn vì các burst có kích thước cố định không thay đổi trong mạng sẽ giúp giảm bớt khả năng mất burst do xung đột (Sự thay đổi chiều dài burst là 0). Bằng cách tính toán giá trị chiều dài burst ngắn nhất, giá trị thời gian định thời dựa trên khả năng chịu trễ của gói ta có thể đạt được xác suất mất burst nhỏ nhất mà vẫn thỏa yêu cầu trễ.

Do lưu lượng trong mạng có thể thay đổi nên hiện nay phương pháp thiết lập burst tốt nhất là vừa thiết lập theo thời gian, vừa theo độ dài burst. Trong cách này, burst sẽ được thiết lập trong một khoảng thời gian nhất định, sau thời gian này các burst sẽ được gởi đi mà không xét đến độ dài của burst do đó các burst sẽ có độ dài khác nhau nhưng không nhỏ hơn độ dài qui định, nếu độ dài burst nhỏ hơn độ dài qui định thì một phần bổ sung sẽ được thêm vào phần burst đó để được độ dài qui định nhỏ nhất. Nếu chưa hết thời gian này mà độ dài burst có giá trị bằng độ dài lớn nhất thì burst sẽ được gởi đi trước khi kết thúc thời gian thiết lập burst.




Hình 1.2 Các phương pháp thiết lập burst theo chiều dài burst và theo thời gian

Trong [3], kỹ thuật thiết lập burst dựa trên dự đoán được được giới thiệu, trong đó giá trị ngưỡng của burst hay gán trị định thời của burst kế tiếp được dự đoán dựa trên tốc độ trung bình của lưu lượng tới. Bằng cách sử dụng chiều dài burst dự đoán, gói BHP có thể được gửi đi vào mạng lõi trước khi một burst thực sự được tạo ra và có thể dự trữ tài nguyên trước đó, do đó có thể làm giảm độ trễ do thiết lập burst. Giá trị dự đoán có thể được sử dụng cho việc thiết lập các giá trị mức ngưỡng hay bộ định thời cho burst kế tiếp dựa trên tính tương quan của lưu lượng. Ưu điểm của phương pháp thiết lập burst dựa trên dự đoán là báo hiệu và thiết lập burst có thể thực hiện song song do đó tiết kiệm được thời gian thiết lập burst.

Trong lúc thiết lập burst, gói đến ở lớp cao hơn được chứa trong hàng đợi dựa trên đích đến và lớp QoS của chúng. Sau khi tiêu chuẩn thiết lập burst được thỏa mãn (mức ngưỡng kích thích burst hay giá trị của bộ định thời đạt được), burst

sẽ được tạo ra và gửi vào mạng. Do đó, chúng ta có thể thấy đặc tính đến của gói và phân phối chiều dài gói ảnh hưởng nhiều đến đặc tính đến của burst và phân phối chiều dài burst.

Trong lúc thiết lập burst, node biên ngõ vào sắp xếp và lập lịch cho các gói đến vào trong những bộ đệm ngõ vào theo mức QoS và đích đến của nó. Những gói này sau đó được tập hợp thành burst và chứa trong các bộ đệm ngõ ra. Bởi vì mỗi hướng và mỗi lớp dịch vụ yêu cầu một bộ đệm riêng, nên số lượng lớp dịch vụ và kích thước mạng quyết định nhiều đến kích thước của bộ đệm tại node biên ngõ vào.

Một tình huống phức tạp hơn khi gói đến có nhiều lớp dịch vụ. trong trường hợp này, các gói đến phải được đóng thành burst cùng với mức ưu tiên của nó vào trong mỗi burst để mạng lõi quang có thể cung cấp các mức dịch vụ khác nhau. Việc lựa chon một cơ cấu thiết lập burst cho tất cả các lớp dịch vụ có thể là không thích hợp. Một phương pháp thiết lập burst dựa trên mức ngưỡng hay bộ định thời với giá trị bộ định thời lớn có thể dẫn đến những độ trễ không chấp nhận được cho các lớp dịch vụ yêu cầu nghiêm ngặt về độ trễ, trong khi chiều dài burst không tối ưu có thể làm tăng độ mất gói đối với các lớp dịch vụ yêu cầu nghiêm ngặt về mất mát dữ liệu. Trong [3] đã nêu lên cách thiết lập burst kết hợp để khắc phục những vấn đề này. Trong phương pháp thiết lập burst kiểu kết hợp, gói từ các lớp dịch vụ khác nhau với những yêu cầu về QoS khác nhau có thể thiết lập trên cùng một burst. Phần mô phỏng quá trình thiết lập burst ở chương 5 sử dụng kỹ thuật thiết lập burst kiểu vi phân hỗ trọ nhiều lớp dịch vụ khác nhau. Trong phương pháp này, loại burst được định nghĩa dựa trên yêu cầu về QoS. Mỗi loại burst sau đó được thiết lập sử dụng một cơ cấu thiết lập thích hợp để chắc chắn rằng đáp ứng được yêu cầu về QoS. Giá trị của bộ định thời dựa trên yêu cầu nghiêm ngặt của độ trễ end-to-end và giá trị của mức ngưỡng được thiết lập bằng giá trị tối ưu của độ dài burst với lưu lượng tải vào mạng nằm trong một dải cho trước.

### 1.5 Thời gian offset

Trong mạng OBS có sự liên kết chặt chẽ về thời gian giữa gói tin điều khiển và burst dữ liệu. Burst được gửi đi ngay sau gói tin điều khiển một thời gian

offset đủ để dự trữ tài nguyên cho burst tại các node trung gian. Thời gian offset này ít nhất phải bằng thời gian xử lý ở các node của gói tin điều khiển. $T_{offset} = \Delta. H + T_{xl} + T_{ch}$ với H là số lượng node chuyển mạch trung gian trên đường truyền và $\Delta$ là thời gian cần thiết để xử lý ở mỗi node. $T_{xl}$ và $T_{ch}$ là thời gian xử lý và chuyển mạch burst ở node đích.

Một yêu cầu đặt ra là phải tính toán sao cho thời gian offset không dài quá hay ngắn. Nếu thời gian offset quá ngắn gây ra tình trạng burst được gửi đi khi chưa hoàn thành dự trữ tài nguyên ở các node trung gian, burst đó sẽ bị mất. Ngược lại nếu thời gian offset quá dài làm chậm trễ quá trình truyền burst trong mạng.

Một cách để xác định đúng thời gian offset là biết được số node mà burst phải truyền qua trên đường truyền. Tuy nhiên, số lượng node trung gian giữa node nguồn và node đích trong mạng OBS thường không biết trước được và nếu có thể biết được thì do lộ trình có thể thay đổi, sự thay đổi này có thể do định tuyến làm lệch khi có xung đột ở các node trung gian, nên nó cũng không thích hợp khi sử dụng. Do đó vấn đề tính thời gian offset cũng là một vấn đề cần thiết trong mạng OBS. Yêu cầu đưa ra là phải có một giá trị offset không phụ thuộc đường truyền và không yêu cầu sự trao đổi thông tin giữa các node. Trên cơ sở độ lớn của thời gian offset, có thể chia thời gian offset  thành các loại.

### 1.5.1 Offset cố định

Offset này được dùng chủ yếu trong giao thức JET, trong đó nó được tính bằng tổng thời gian xử lý gói tin điều khiển ở các node trung gian và node đích cũng như thời gian cấu hình chuyển mạch ở node đích. Với các chuyển mạch tốc độ cao thì có thể giả thiết thời gian xử lý gói tin điều khiển ở các node trung gian là khá nhỏ nên thời gian offset được tính là thời gian xử lý gói tin điều khiển và cấu hình chuyển mạch ở node đích. Ta có thể lấy giá trị lớn nhất trong các thời gian offset tính ở các node đích để làm thời gian offset chung cho toàn mạng. Thời gian offset không phụ thuộc đường truyền làm đơn giản hóa việc tính toán và thực thi các giao thức báo hiệu trong mạng chuyển mạch burst quang.

### 1.5.2 Offset khi không có sự dự trữ

Trong kiểu offset này burst được gởi đi ngay sau gói tin điều khiển. Thời gian offset này được tính bằng thời gian truyền của gói tin điều khiển. Thời gian offset này chỉ được áp dụng trong mạng có thời gian thiết lập chuyển mạch cũng như xử lý chuyển mạch là rất ngắn.

## 1.6 Kết luận chương

Qua những nội dung đã trình bày trong chương này giúp ta có được cái nhìn tổng quan về công nghệ chuyển mạch chùm quang OBS, các tính năng vượt trội của cũng như khả năng ứng dụng trong thực tế của OBS so với các công nghệ khác như chuyển mạch kênh quang hay chuyển mạch gói quang. Các phương pháp thiết lập burst dựa trên mức ngưỡng về độ dài burst hay bộ định thời cũng được giới thiệu từ đó đề xuất phương pháp thiết lập burst nhằm mục đích giảm thiểu sự mất burst. Mặc dù chưa được biết đến nhiều như chuyển mạch kênh quang và chuyển mạch gói quang nhưng chuyển mạch chùm quang OBS với những tính năng ưu việt hứa hẹn sẽ trở thành công nghệ chuyển mạch cho tương lai, là giải pháp hiệu quả cho mạng đường trục thế hệ mới.

# Chương 2

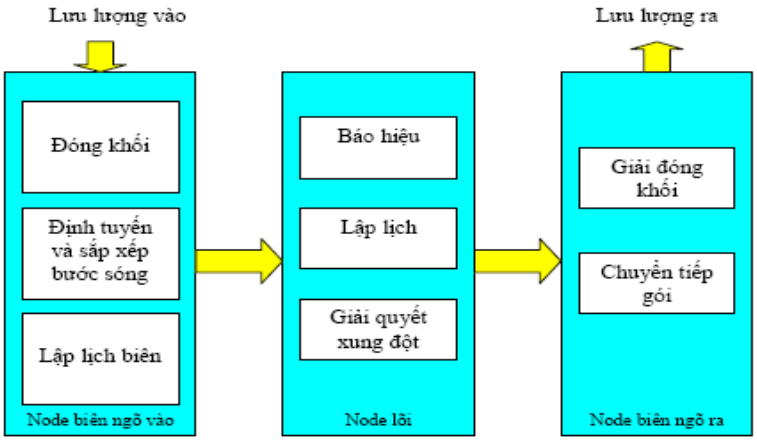
# KIẾN TRÚC MẠNG CHUYỂN MẠCH CHÙM QUANG OBS

## 2.1. Giới thiệu chương

Cấu trúc phần cứng là một phần quan trọng trong OBS, nó làm cho OBS có các chức năng riêng cũng như có những ưu điểm hơn so với các chuyển mạch khác. Chương này giới thiệu về cấu trúc của chuyển mạch OBS gồm các nội dung chính như giới thiệu mạng OBS ở dạng mắt lưới hay dạng vòng node,cấu trúc của node biên, node lõi. Trong mạng OBS, các gói IP khác nhau được tập hợp thành các burst ở node biên đầu vào sau đó được truyền đi, các gói IP được kết hợp này được tách rời trở lại ở node biên đầu ra. Chức năng tạo burst bởi sự kết hợp và giải kết hợp được thực hiện khác nhau như có thể sử dụng một ngưỡng hoặc khoảng thời gian quy định để kết hợp các gói dữ liệu tạo ra một burst quang và gửi burst vào mạng. Các node lõi sẽ có các bộ thu WDM, các bộ phát WDM, các bộ ghép kênh WDM, các bộ giải ghép kênh WDM các bộ khuyếch đại node, các đơn vị điều khiển chuyển mạch, các bộ biến đổi bước sóng, các đường tạo trễ, các bộ chuyển mạch phân chia không gian. Như vậy node biên và node lõi phải có cấu trúc phù hợp để thực hiện các chức năng của nó được trình bày ở các phần sau.

## 2.2. Kiến trúc của mạng OBS

Hình 2.1 Mô tả thành phần của mạng OBS với các chức năng khác nhau



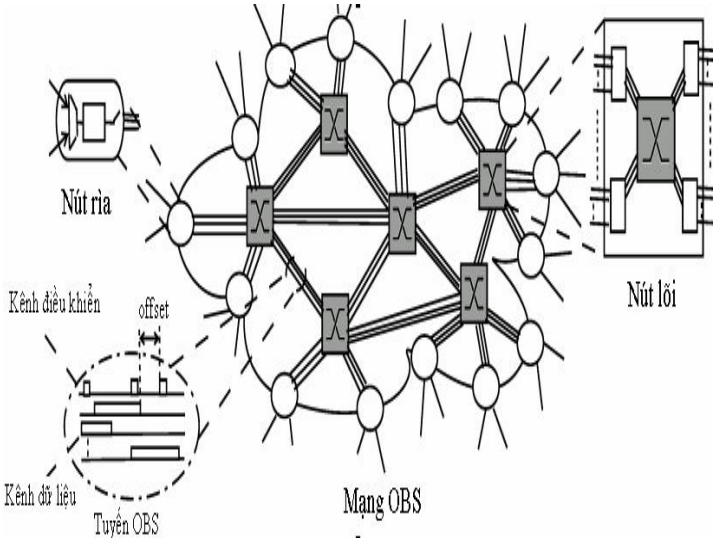

Hình 2.1. Sơ đồ các chức năng của mạng OBS

Trong mạng OBS, mỗi node có thể hỗ trợ hai loại lưu lượng cả điện lẫn quang. Do đó, mỗi node bao gồm một node lõi và một node biên, ta gọi node này là node kết hợp.

### 2.2.1 Kiến trúc mạng OBS dạng mắt lưới

Trong mạng chuyển mạch burst quang các burst dữ liệu bao gồm tổ hợp nhiều gói được chuyển qua mỗi node mạng ở dạng toàn quang. Một thông báo điều khiển được truyền trước burst dữ liệu với mục đích thiết lập các chuyển mạch dọc theo đường đi của burst. Burst dữ liệu được truyền theo sau gói điều khiển mà không đợi báo nhận để thiết lập kết nối.

Hình 2.2 thể hiện một mạng OBS dạng mắt lưới bao gồm các node biên và các node lõi. Mạng OBS bao gồm các chuyển mạch burst quang được nối với các tuyến WDM. OBS phát một burst từ cổng đầu vào tới cổng đầu ra, dựa trên thiết kế chuyển mạch nó có thể có hoặc không được trang bị bộ đệm quang. Các tuyến WDM mang tổ hợp nhiều bước sóng và mỗi bước sóng coi như một kênh truyền. Gói điều khiển kết hợp với một burst cũng có thể truyền trên băng tần qua cùng một kênh như là dữ liệu, hoặc trên một kênh điều khiển riêng biệt. Burst có thể được cố định để mang một hoặc nhiều gói IP.



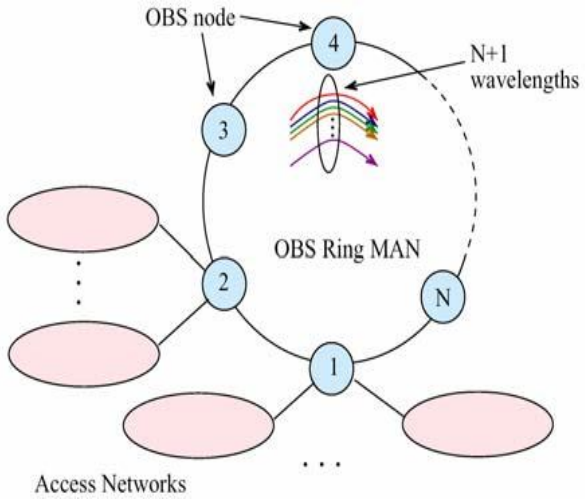

Hình 2.2. Mô hình mạng OBS dạng mắt lưới

Một node chuyển mạch đặc trưng bao gồm những thành phần sau:

- Giao diện đầu vào: Tiếp nhận gói điều khiển và burst dữ liệu, chuyển đổi gói điều khiển thành tín hiệu điện.
- Đơn vị điều khiển chuyển mạch: Phiên dịch gói điều khiển, đặt lịch trình và giải quyết xung đột, định tuyến, điều khiển ma trận chuyển mạch, tạo lại gói mào đầu và điều khiển biến đổi bước sóng.
- Các bộ biến đổi bước sóng và các đường trễ quang (ODL): đường trễ quang sử dụng như một bộ đệm để chứa burst trong một khoảng thời gian trễ nhất định.
- Đơn vị chuyển mạch quang: Các chuyển mạch không gian làm nhiệm vụ chuyển burst dữ liệu.

### 2.2.2. Kiến trúc mạng OBS dạng Vòng và Node

Chúng ta xem xét mạng gồm N node OBS được tổ chức trong một vòng Ring đơn hướng, như trên hình 2.3




Hình 2.3. Mô hình mạng OBS dạng vòng RING

Mỗi sợi kết nối giữa hai node OBS liên tiếp trong vòng ring có thể hỗ trợ N+1 bước sóng. Trong đó N bước sóng được sử dụng để truyền burst, bước sóng thứ N+1 được sử dụng như một kênh điều khiển.

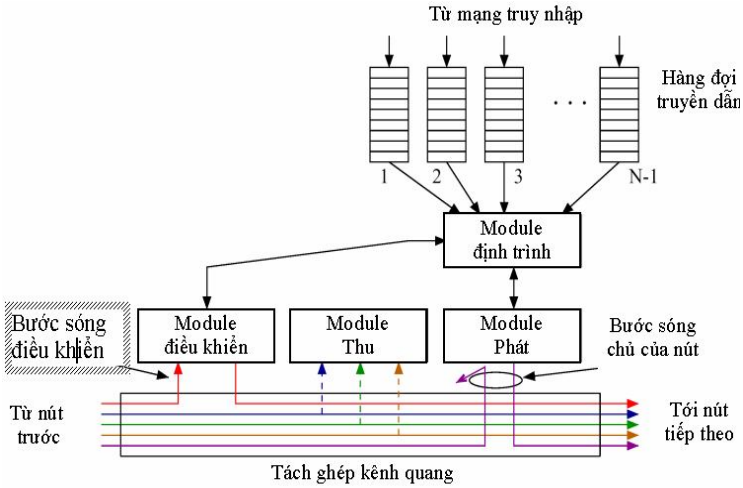
Mỗi node OBS được gắn với một hoặc nhiều mạng truy nhập. Theo chiều

từ mạng truy nhập đến vòng Ring, các node OBS hoạt động như một bộ tập trung. Dữ liệu từ người sử dụng cần chuyển qua mạng Ring được tập hợp, lưu trữ (đệm) ở dạng điện tử rồi sau đó được nhóm lại cùng nhau và được truyền trong burst tới node OBS đích. Mỗi burst có thể có kích thước bất kỳ giữa giá trị cực đại và cực tiểu. Các burst được truyền đi ở dạng tín hiệu quang dọc theo vòng Ring mà không trải qua bất kỳ sự chuyển đổi điện-quang nào ở những node trung gian.

Theo hướng từ vòng Ring đến các mạng truy nhập, node OBS ngắt các burst quang đã được định sẵn tới chính nó, chuyển tín hiệu quang thành tín hiệu điện tử, xử lý điện tử dữ liệu chứa đựng trong burst và chuyển giao chúng tới những người dùng trong các mạng truy nhập gắn liền với nó.

Kiến trúc của một node OBS được cho thấy trong hình 2.4, mỗi node được trang bị một bộ tách ghép kênh quang (OADM), và hai cặp thu phát quang. Cặp đầu tiên gồm có một máy thu và máy phát cố định được điều hưởng bởi bước sóng điều khiển, và là bộ phận của module điều khiển.

Bước sóng điều khiển được tách bởi OADM ở mỗi node, và được ghép trở lại sau khi module điều khiển đã đọc thông tin điều khiển và có thể chèn thông tin mới vào. Cặp thứ hai của bộ phận thu và phát gồm có một máy phát được cố định để điều hưởng tới bước sóng chủ của node, và một máy thu nhanh để có thể nhận các burst từ tất cả N bước sóng truyền tới. Mỗi node OBS có một bước sóng chủ chuyên dụng để truyền các burst của chính nó. Bộ OADM ở mỗi node loại bỏ tín hiệu quang từ bước sóng chủ của node bằng cách tách bước sóng tương ứng. Bộ OADM cũng tách tín hiệu quang trên những bước sóng khác nhau, mỗi khi các bước sóng đó chứa đựng các burst cho node này.

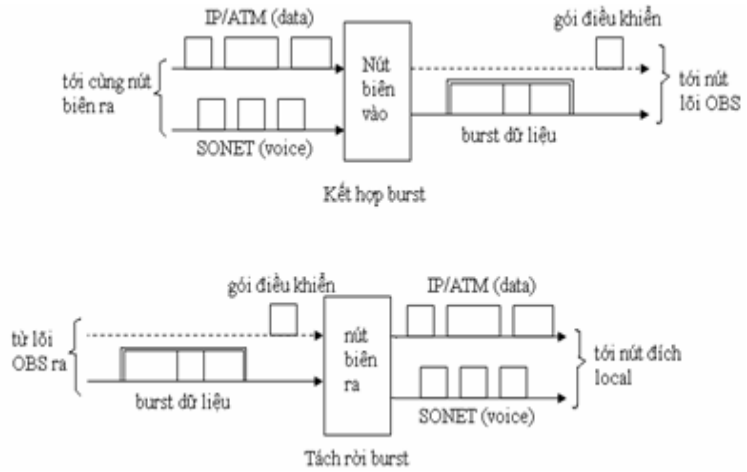



Hình 2.4. Kiến trúc node chuyển mạch quang

Trong trường hợp khi có nhiều burst đến, mỗi burst trên một bước sóng khác nhau, ở một node OBS, module thu trong hình 2.4 sử dụng chiến lược giải quyết xung đột để xác định burst nào sẽ được chấp nhận. Dữ liệu truyền đi được tổ chức thành những hàng đợi truyền dựa theo đích của chúng. Bộ đệm dữ liệu ở mỗi node OBS được chia sẽ thành N-1 hàng đợi. Mỗi hàng đợi tương ứng với một trong số N-1 node đích. Các hàng đợi phục vụ theo thứ tự xác định bởi module lịch trình như trong hình 2.4.

### 2.2.3. Cấu trúc và chức năng của node biên

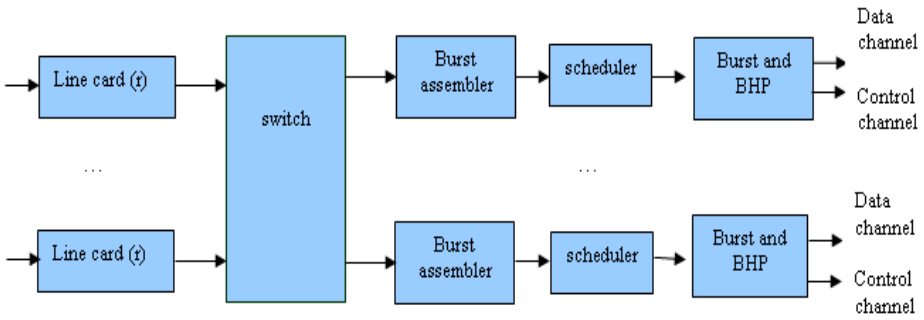
Node biên OBS gồm hai loại, node biên ở đầu vào và node biên ở đầu ra, đây là giao diện giữa mạng IP và mạng OBS. Trong mạng OBS, các gói IP khác nhau được tập hợp thành các burst tại node biên đầu vào sau đó nó được truyền đi, và các IP đã được kết hợp lại này sẽ được tách rời trở lại tại node biên đầu ra, quá trình này được thể hiện như sau:




Hình 2.5. Kết hợp và tách rời burst trong mạng OBS

Tương ứng với mỗi burst gói tin điều khiển được tạo ra. Gói tin điều khiển mang thông tin như chiều dài burst, thời gian đến của burst, thông tin về node đích và được gởi trên kênh điều khiển có bước sóng dành riêng còn burst được gởi đi trên các kênh dữ liệu. Sự phân chia xử lý này làm cho kênh điều khiển có thể hoạt động ở tốc độ bit thấp hơn so với kênh dữ liệu nên có thể sử dụng các phương pháp điều chế khác nhau. Vì một gói điều khiển nhỏ hơn nhiều so với một burst dữ liệu nên một kênh điều khiển thường mang hàng trăm gói điều khiển, tương ứng với hàng trăm burst dữ liệu.

Cấu trúc cơ bản của node biên đầu vào như hình:

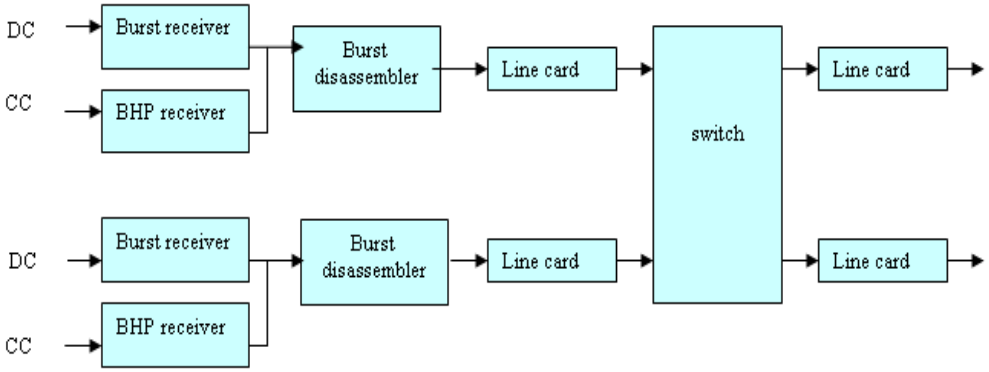



Hình 2.6 Cấu trúc của node biên đầu vào

Ở node biên đầu vào, burst được thiết lập từ các gói tin sau đó được đưa đến bộ sắp xếp chọn đường ra cho burst và truyền trên đường truyền. Gói tin điều

khiển cũng được truyền đi trên kênh bước sóng riêng. Ở node biên đầu ra, các kênh dữ liệu DCG (data channel group) được đưa đến bộ nhận burst. Các kênh điều khiển CCG (control channel group) được đưa đến bộ nhận gói tin điều khiển (BHP receiver).

Cấu trúc cơ bản của node biên đầu ra như hình:




Hình 2.7. Cấu trúc của node biên đầu ra

Tại node biên đầu ra burst được đưa đến bộ tách burst để tách thành các gói tin ban đầu, sau đó được đưa đến chuyển mạch để chuyển mạch đến cổng ra theo yêu cầu.

Trong node biên đầu vào, khối chức năng chính là bộ phát với bước sóng điều chỉnh được còn ở node biên đầu ra là bộ nhận burst.

**Bộ phát với bước sóng điều chỉnh được (Fast Tunable Laser)**

Đây là thành phần chính trong OBS node biên đầu vào, nó cho phép điều chỉnh bước sóng trong một băng thông bước sóng cho trước để truyền burst trên các bước sóng khác nhau. Các bộ phát bước sóng có thể sử dụng kết hợp với bộ chuyển đổi bước sóng giúp cho việc giải quyết hiệu quả xung đột trong mạng OBS.

**Bộ nhận burst (Burst receiver)**

Bộ nhận burst có nhiệm vụ lấy lại thông tin về chiều dài burst và bù lại những suy hao trên đường truyền. Bộ nhận burst phải có khả năng xử lý các thông số sau:

- Cấu trúc burst khác nhau: do có nhiều phương pháp điều chế khác nhau nên có thể có các cấu trúc burst khác nhau. Các hệ thống truyền dẫn hiện có dựa trên

phương pháp điều chế NRZ (non-return to zero) và sự phát hiện trực tiếp tín hiệu dữ liệu. Nếu sử dụng phương pháp điều chế DPSK (differential PSK) thì độ nhạy cao hơn và thích hợp với nhiều loại kiến trúc vật lý hơn. Trong mạng OBS đều có thể sử dụng các phương pháp điều chế này nên burst receiver phải có khả năng xử lý các cấu trúc riêng của mỗi phương pháp.

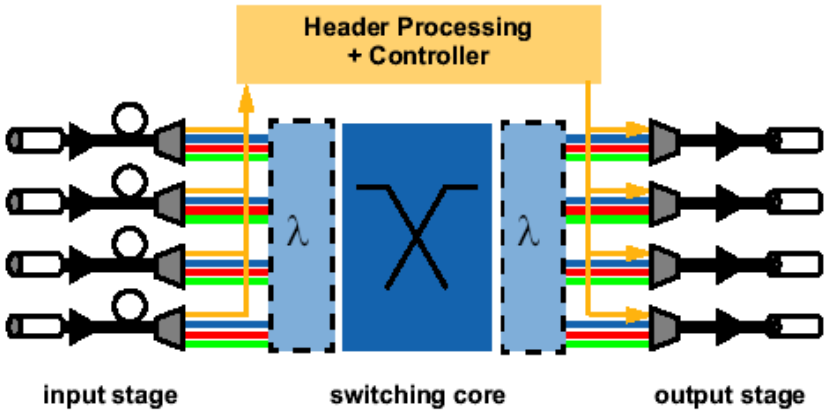
- Sự thay đổi độ dài burst: các burst trong OBS có thể có độ dài khác nhau nên bộ thu cần phải đồng bộ với mọi burst đến.
- Sự thay đổi của khoảng hở giữa các burst: Bộ thu phải có khả năng nhận được burst đơn sau một khoảng hở giữa các burst. Nếu bộ thu có thể hoạt động trong chế độ truyền liên tục thì không cần quan tâm đến khoảng hở giữa các burst khi và chỉ khi các burst đến bộ thu từ một bộ phát. Nếu burst gởi đi từ các bộ phát khác nhau thì sự đồng bộ khoảng hở giữa các burst là cần thiết.
- Công suất burst thay đổi: các burst khác nhau được khuếch đại và suy hao khác nhau trên đường truyền do đó bộ thu phải có khả năng đáp ứng đối với các mức công suất khác nhau của các burst khác nhau.

### 2.2.4. Cấu trúc và chức năng của node lõi

Node lõi: cơ bản bao gồm một bộ kết nối chéo quang OXC và một đơn vị điều khiển chuyển mạch SUC. SUC tạo và bảo trì một bảng chuyển tiếp và chịu trách nhiệm cấu hình cho OXC. Khi SUC nhận một gói BHP, nó đọc thông tin trong gói xác định đích của gói này và burst dữ liệu theo sau, tra cứu thông tin trong bảng chuyển tiếp để đưa ra quyết định nên mở ngõ ra nào của khối kết nối chéo quang OXC. Nếu ngõ ra có thể sử dụng khi được khi burst dữ liệu đến, SCU sẽ cấu hình cho phép burst dữ liệu chuyển thẳng sang hoàn toàn quang. Nếu ngõ ra mong muốn không thể sử dụng tức đang được sử dụng bởi một burst khác, việc cấu hình cho OXC phụ thuộc vào nguyên tắc giải quyết xung đột được đưa vào mạng. Nói chung, SUC chịu trách nhiệm đọc các gói điều khiển, lập lịch, nhận biết xung đột và giải quyết xung đột, tra cứu bảng chuyển tiếp, điều khiển ma trận chuyển mạch (hay OXC), tạo lại gói điều khiển để phát tiếp nếu node này chưa phải là đích của nó và

điều khiển việc chuyển đổi bước sóng. Trường hợp một burst dữ liệu vào OXC trước gói điều khiển của nó thì burst này sẽ bị rớt.

Cấu trúc chung của OBS core node gồm các khối chính: đơn vị điều khiển chuyển mạch O/E/O, cơ cấu chuyển mạch và bộ chuyển đổi bước sóng.




Hình 2.8. Cấu tạo của node lõi trong mạng OBS

- Đơn vị điều khiển chuyển mạch với bộ chuyển đổi O/E/O

Đơn vị điều khiển chuyển mạch có chức năng xử lý gói tin điều khiển, lấy ra các thông tin định tuyến và bước sóng, điều khiển cơ cấu chuyển mạch và bộ chuyển đổi bước sóng để chuyển burst đến cổng ra mong muốn trên kênh bước sóng mong muốn. Trong OBS, gói tin điều khiển được xử lý trong miền điện nên bộ chuyển đổi điện quang là cần thiết.

- Cơ cấu chuyển mạch quang

Cơ cấu chuyển mạch quang thường sử dụng chuyển mạch không gian quang. Do node có N đầu vào và M bước sóng trên mỗi cáp quang nên phải sử dụng chuyển mạch không gian NM x MN. Trong OBS, cơ cấu chuyển mạch quang phải có kích thước lớn, thời gian chuyển mạch nhanh, có độ tin cậy cao và chi phí thấp để giảm chi phí trong mạng do trong OBS phải sử dụng bộ chuyển đổi O/E/O, bộ chuyển đổi bước sóng và có thể sử dụng các đường dây trễ nên rất tốn kém.

- Khối chuyển đổi bước sóng

Khối chuyển đổi bước sóng có thể đặt ở đầu vào hay đầu ra của cơ cấu chuyển mạch. Nếu bộ chuyển đổi bước sóng đặt ở đầu ra thì các bước sóng ở đầu ra của bộ chuyển đổi không đổi, nếu đặt ở đầu vào thì bước sóng đầu ra có thể thay đổi. Trong hai trường hợp trên thì vẫn có thể có được sự chuyển đổi bước sóng đầy đủ nếu sử dụng bộ chuyển đổi bước sóng hoàn toàn.

Để giảm sự phức tạp và tốn kém của chuyển mạch, các node OBS có thể chia sẻ các bộ chuyển đổi bước sóng. Tuy nhiên, nếu dùng chung nó chỉ có thể chuyển đổi một số hạn chế các bước sóng và việc thực hiện công nghệ này phải được tính toán kỹ. Nếu muốn chuyển đổi toàn bộ bước sóng với chuyển mạch này thì cấu trúc chuyển mạch càng phức tạp hơn. Chuyển đổi bước sóng toàn bộ là cần thiết để giải quyết xung đột trong OBS nên ở đây chỉ đề cập đến chuyển mạch với bộ chuyển đổi bước sóng toàn bộ.

Tóm lại node biên đầu vào có chức năng thiết lập burst,định tuyến,gán bước sóng và sắp xếp burst tại biên đầu vào.Các node lõi có chức năng báo hiệu, sắp xếp burst tại các liên kết trong lõi và giải quyết xung đột. Các node biên đầu ra chịu trách nhiệm tách burst thành các gói riêng rẽ rồi truyền đến lớp mạng cao hơn.

## 2.3 Kết luận chương

Như vậy chương này đã trình bày được cơ bản cấu trúc phần cứng và sơ đồ chức năng của mạng OBS thể hiện được ưu điểm nổi trội của nó so với các chuyển mạch khác. Đặc biệt chú trọng vào cấu trúc của node biên đầu vào, node biên đầu ra và node lõi để thực hiện các chức năng kết hợp burst ở đầu vào và giải kết hợp burst ở đầu ra, việc xử lý burst, cấp phát bước sóng, khuếch đại bước sóng,… của node lõi. Ngoài ra mạng OBS bao gồm các chuyển mạch burst quang được nối bởi các tuyến WDM, các tuyến WDM này mang tổ hợp các bước sóng và mỗi bước sóng coi như một kênh truyền. Gói kênh điều khiển kết hợp với một burst được truyền trên kênh điều khiển riêng biệt hoặc trên cùng kênh như là kênh dữ liệu. Hiểu được cấu trúc phần cứng để thấy được các ưu điểm của chuyển mạch OBS và khai thác các ưu điểm đó trong việc đáp ứng nhu cầu truyền dữ liệu là một việc hết sức quan trọng.

**Chương 3**

# BÁO HIỆU VÀ GIẢI QUYẾT XUNG ĐỘT TRONG MẠNG OBS

## 3.1 Giới thiệu chương

Khi một burst được gửi tới node lõi, tiến trình báo hiệu được tiến hành để dự trữ tài nguyên và cấu hình cho bộ chuyển mạch quang tại mỗi node. Tiến trình báo hiệu trong mạng chuyển mạch burst quang thực hiện trên các gói header và các gói này được truyền độc lập với các burst dữ liệu. Bên cạnh đó để tăng hiệu quả truyền dữ liệu, giảm khả năng mất burst trong mạng OBS ta phải có các phương pháp giải quyết xung đột thích hợp. Trong chương này, em sẽ trình bày các thông số và các tính chất khác nhau của các giao thức báo hiệu trong mạng OBS cũng như đặc điểm riêng của từng phương thức giải quyết xung đột trong mạng OBS.

## 3.2. Báo hiệu trong mạng OBS

Trong mạng OBS gói tin header được truyền trên một bước sóng khác với bước sóng của burst dữ liệu tương ứng với nó. Header đi cùng đường và tới các node trước burst dữ liệu, tại các node này header cung cấp thông tin cho các node cấu hình bộ kết nối chéo quang sao cho phù hợp với thời gian tới tương ứng của burst dữ liệu.

### 3.2.1. Phân loại các giao thức báo hiệu

Có nhiều loại giao thức báo hiệu dùng cho chuyển mạch burst quang, tùy vào cách thức và thời điểm mà tài nguyên dọc theo tuyến truyền được dự trữ cho một burst. Cụ thể, phương pháp báo hiệu có thể được phân loại bởi các tính chất sau:

- Dự trữ 1 chiều (one-way reservastion), dự trữ hai chiều (two-way reservation), hay kết hợp.
- Khởi tạo tại node nguồn (source-initiated), node đích (destination-initiated), hay node trung gian (intermediate-node-initiated reservation).
- Dự trữ liên tục hay không liên tục.
- Dự trữ tức thời hay có trì hoãn.

- Giải phóng tài nguyên tường minh không tường minh.
- Báo hiệu tập trung hay phân bố

#### 3.2.1.1 Phương thức dự trữ một chiều, hai chiều hay kết hợp

Dựa vào cách hoạt động của phương pháp báo hiệu, ta phân làm 3 loại: dự trữ một chiều (one-way reservation), dự trữ hai chiều (two-way reservation), và dự trữ kết hợp (hybrid reservation).

Ở báo hiệu dùng các dự trữ một chiều, node nguồn gửi ra một gói điều khiển yêu cầu mỗi node dọc trên tuyến đường cấp phát tài nguyên cần thiết cho burst dữ liệu và cấu hình kết nối chéo ở các node cho phù hợp. Tiếp theo node nguồn gửi ra burst dữ liệu mà không chờ bản tin ACK từ các node trung gian hay node đích, mặc cho việc dự trữ tài nguyên ở các node là thành công hay thất bại. Vì việc dự trữ không được xác nhận (một chiều), nên burst dữ liệu có thể bị drop. Tuy nhiên, vì không phải chờ bản tin ACK báo về nên burst dữ liệu được gửi ra sớm hơn, giảm được độ trễ khi truyền dữ liệu từ đầu cuối tới đầu cuối.

Phương pháp báo hiệu dự trữ hai chiều dựa vào bản tin ACK. Khi header được gửi ra từ node nguồn tới node đích để dự trữ tài nguyên cho một burst dữ liệu thì có một bản tin ACK được gửi ngược trở lại, xác nhận rằng tài nguyên yêu cầu đã được cấp phát thành công. Burst dữ liệu chỉ được truyền sau khi nhận được bản tin ACK. Nếu bất kì một node trung gian nào dọc trên đường truyền không thể tiếp nhận được burst dữ liệu thì chính tại node gây ra gián đoạn đó sẽ gửi bản tin NACK (Negative Acknowledgement) về node nguồn, báo rằng việc dự trữ đã thất bại. Node này cũng sẽ thực hiện những hoạt động thích hợp để giải phóng tất cả các dự trữ (nếu có) trên các link phía trước của đường truyền. Phía nguồn có thể chọn cách thực hiện yêu cầu dự trữ lại bằng cách gửi đi một header mới, hay cho drop luôn yêu cầu đó. Phương pháp báo hiệu có xác nhận (hai chiều) việc dự trữ có thể giảm thiểu khả năng mất burst dữ liệu trong mạng lõi OBS nhưng nó lại gây ra độ trễ lớn hơn cho mỗi burst khi truyền từ đầu cuối tới đầu cuối.

Phương pháp báo hiệu kết hợp đưa ra giải pháp cân bằng giữa dự trữ một chiều và hai chiều, đây là phương pháp có một phần xác nhận việc dự trữ. Trong phương pháp báo hiệu kết hợp, việc dự trữ từ node nguồn tới các node trung gian trên tuyến đường được xác nhận bằng bản tin ACK, trong khi việc dự trữ từ node trung gian tới đích thì không được xác nhận. Vị trí của node được chỉ làm node trung gian sẽ xác định khả năng mất hay độ trễ của burst dữ liệu. Nếu node trung gian gần với nguồn thì hoạt động của mạng sẽ giống như việc dự trữ không có xác nhận (một chiều), và nếu node trung gian gần về phía đích thì hoạt động giống như việc dự trữ có xác nhận (hai chiều).

#### 3.2.1.2 Phương thức dự trữ được khởi tạo ở node nguồn, node đích và ở node trung gian

Một giao thức báo hiệu có thể khởi tạo yêu cầu dự trữ tài nguyên tại nguồn, đích hay tại một bước trung gian nào đó. Trong phương pháp dự trữ được khởi tạo tại node nguồn (source initiated reservation – SIR), tài nguyên cho burst dữ liệu được dự trữ theo đường xuôi theo header khi header đi từ nguồn tới đích. Nếu việc cấp phát tài nguyên theo hướng xuôi như thế thành công và một giao thức dự trữ trước tương ứng được dùng thì một bản tin ACK chỉ ra các bước sóng đã được giành trước sẽ được gửi ngược trở về phía nguồn. Tại nguồn, khi nhận được các thông tin về tài nguyên, nó phát burst dữ liệu vào mạng lõi vào thời điểm đã được định trước.

Trong phương pháp dự trữ được khởi tạo ở node đích (Destination Initiated Reservation – DIR ), node nguồn phát ra một yêu cầu về tài nguyên về phía node đích, yêu cầu này thu thập thông tin về các bước sóng đang sẵn sàng trên mỗi link dọc theo tuyến đường. Dựa trên thông tin thu thập được, node đích sẽ chọn ra một bước sóng đang sẵn sàng (nếu tồn tại) và phù hợp với thời điểm tới, tiếp đó nó gửi một yêu cầu dự trữ trước ngược về node nguồn. Yêu cầu dự trữ này sẽ đi qua các node trung gian, thực hiện việc dự trữ các bước sóng đã được chọn trong khoảng thời gian thích hợp. Nguyên nhân chính dẫn tới nghẽn (hay mất dữ liệu) trong SIR

là do thiếu tài nguyên rỗi, trong khi trong DIR**,** mất mát là do thông tin cung cấp lỗi thời, không còn đúng nữa.

Trong phương pháp dự trữ được khởi tạo ở node trung gian (intermediate node initiated reservation - INI), cơ bản nó giống như phương pháp dự trữ tài nguyên DIR trong đoạn từ nguồn tới một node trung gian nào đó, và giống với phương pháp SIR trong đoạn từ node trung gian đó tới node đích.

Nhìn chung, để giảm mất mát tại các node trên hướng xuôi, phương pháp SIR có thể dự trữ nhiều hơn 1 bước sóng (hay tất cả nếu sẵn sàng) khi tới đích, và giải tỏa các dự trữ không cần thiết trên hướng ngược lại. Dùng phương pháp này có thể dẫn tới mạng hoạt động chậm do nghẽn trên hướng xuôi vì thiếu tài nguyên. Trong khi đó, phương pháp DIR chỉ thu thập thông tin về trạng thái hiện thời của các node trung gian rồi mới dựa trên thông tin đó chọn ra bước sóng. Vì vậy thông tin nhận được về trạng thái riêng của từng node không được cập nhật, điều này sẽ dẫn tới việc bước sóng được chọn có thể đã bị lấy đi bởi một yêu cầu khác trong khoảng thời gian từ khi trạng thái của node được thu thập cho tới khi bản tin dự trữ đến được node đó, khoảng thời gian đó gọi là khoảng thời gian "dễ bị xâm nhập" – vulnerable period. Qua đó ta thấy rằng, phương pháp DIR chịu mất mát là do thông tin lỗi thời trong suốt khoảng thời gian vulnerable.

#### 3.2.1.3 Phương thức bền (Persistent) hay không bền (Non-persistent)

Một quyết định mà phương pháp báo hiệu nào cũng phải thực hiện là hoặc chờ đợi tài nguyên bị nghẽn (cho tới khi rỗi) hoặc là chỉ ngay ra rằng có nghẽn và khởi tạo một phương pháp giải quyết phù hợp tránh cho kết nối thất bại như phát lại, chọn đường khác hay đệm lại.

Phương pháp persistent dùng cách chờ nguồn tài nguyên bị nghẽn (cho tới khi hết nghẽn), với các bộ đệm thích hợp được đặt tại các node (node biên và node lõi) để lưu trữ lại các burst đến.

Phương pháp non-persistent mong muốn một giới hạn về độ trễ (tối thiểu khoảng thời gian trễ do round trip), vì vậy một node tuyên bố rằng yêu cầu đã thất

bại nếu tài nguyên không sẵn sàng ngay tức thời và sẽ thực hiện các giao thức giải quyết nghẽn phù hợp.

#### 3.2.1.4 Dự trữ tức thời (Intermediate Reservation) hay dự trữ có trì hoãn (Delayed Reservation)

Dựa vào khoảng thời gian mà kênh bị dự trữ, các phương pháp báo hiệu được phân thành loại dự trữ tức thời hay dự trữ có trì hoãn.

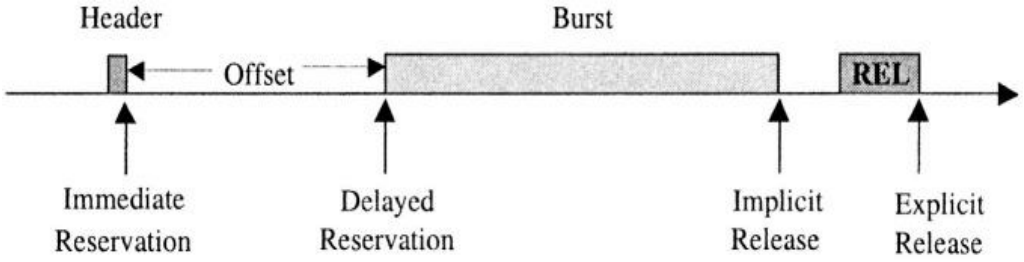
Trong phương pháp dự trữ tức thời, kênh truyền được dự trữ ngay khi bản tin thiết lập (header) đến được node. Trong khi đó, ở phương pháp dự trữ có trì hoãn thì kênh truyền được dự trữ lúc burst dữ liệu thật sự tới node (hay link). Để thực hiện việc dự trữ có trì hoãn, header phải mang thông tin của offset time giữa header này với burst dữ liệu tương ứng với nó. Ví dụ như trong phương pháp báo hiệu just-in-time (JIT), dùng cách dự trữ tức thời, còn phương pháp báo hiệu just-enough-time (JET) dùng cách dự trữ có trì hoãn. Nhìn chung, dự trữ tức thời đơn giản và thiết thực khi thực hiện, nhưng khả năng gây nghẽn cao hơn vì cấp phát băng thông không hiệu quả. Trong khi đó, thực hiện dự trữ có trì hoãn lại phải liên quan tới nhiều thứ hơn nhưng tận dụng băng thông kênh truyền tốt hơn. Phương pháp dự trữ có trì hoãn còn làm phát sinh khoảng trống không làm gì ở giữa các burst được sắp xếp trên kênh dữ liệu. Các giải thuật sắp xếp được sử dụng trong quá trình dự trữ sẽ lưu trữ thêm thông tin về khoảng trống. Dựa vào thông tin đó, bộ scheduler sẽ cấp phát một bước sóng cho yêu cầu dự trữ.

#### 3.2.1.5 Giải tỏa tường minh (Explicit Release) hay không tường minh (Implicit Release)

Một dự trữ có thể được giải tỏa bằng hai cách, tường minh hoặc không tường minh. Trong phương pháp giải tỏa tường minh, một bản tin điều khiển riêng sẽ được gửi theo burst dữ liệu từ nguồn tới đích để giải tỏa hay hủy một dự trữ đang tồn tại. Trong khi đó, trong phương pháp giải tỏa không tường minh, header phải mang thêm thông tin chẳng hạn như thông tin về chiều dài burst và offset time. Ta có thể thấy phương pháp giải tỏa không tường minh cho kết quả tốt hơn trong hoạt động tránh mất dữ liệu vì không có độ trễ giữa thời điểm kết thúc thật sự của burst

dữ liệu và thời điểm đến của bản tin điều khiển giải tỏa tại mỗi node. Trong khi đó, phương pháp giải tỏa tường minh cho kết quả tận dụng băng thông thấp hơn và gia tăng độ phức tạp của bản tin.

Dựa trên giao thức dự trữ và giải tỏa tài nguyên, các phương pháp báo hiệu có thể được chia ra thành 4 loại: Dự trữ tức thời với giải tỏa tường minh, dự trữ tức thời với giải tỏa không tường minh, dự trữ có trì hoãn với giải tỏa tường minh, dự trữ có trì hoãn với giải tỏa không tường minh.




Hình 3.1: Các phương pháp dự trữ và giải tỏa trong mạng OBS.

Dự trữ tức thời và giải tỏa tường minh đòi hỏi có một bản tin điều khiển rõ ràng được gửi đi để thực thi chức năng đã định trước, ví dụ như dự trữ kênh truyền hay giải tỏa một kết nối. Trong phương pháp dự trữ có trì hoãn, header out-of-band cần mang thông tin về offset time, và nếu là giải tỏa không tường minh thì mang thêm thông tin về chiều dài của burst dữ liệu. Ta có thể dễ dàng thấy được phương pháp dùng cách dự trữ có trì hoãn và giải tỏa không tường minh cho kết quả tận dụng hiệu quả băng thông cao hơn, trong khi phương pháp dự trữ tức thời và giải tỏa tường minh tuy thực hiện đơn giản nhưng hiệu quả tận dụng băng thông thấp hơn.

#### 3.2.1.6 Báo hiệu tập trung hay phân bố

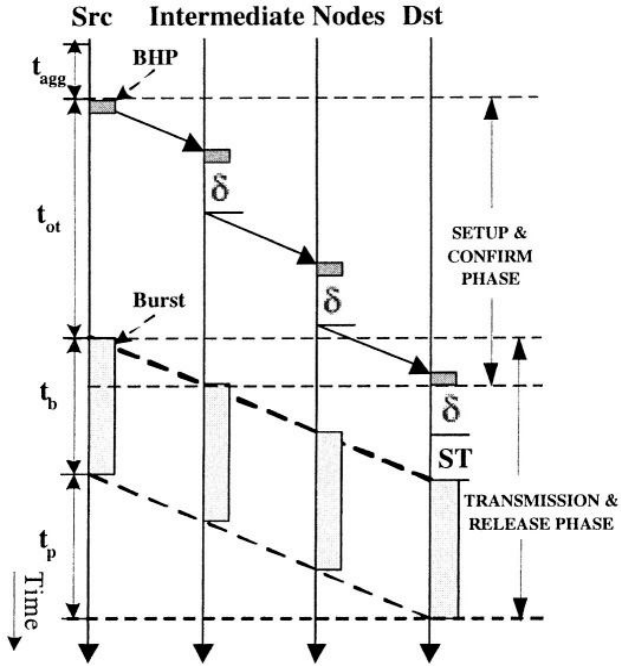
Trong giao thức báo hiệu tập trung, một server được giành riêng để tập trung giải quyết các yêu cầu dự trữ, nó thực hiện nhiệm vụ thiết lập tuyến đường và cấp phát bước sóng trên mỗi tuyến cho mỗi burst dữ liệu đối với tất cả các đôi node nguồn-đích trong mạng. Giao thức tập trung này có thể thực thi có hiệu quả trong mạng nhỏ và lưu lượng không đột biến. Mặt khác, trong giao thức báo hiệu phân tán, mỗi node đều có một bộ scheduler burst riêng, thực hiện nhiệm vụ cấp phát

kênh xuất cho mỗi header đến theo kiểu phân phối. Phương pháp phân phối thích hợp với mạng quang lớn và lưu lượng dữ liệu đột biến.

Hai phương pháp báo hiệu nổi bật trong mạng không dùng bộ đệm OBS là Tell-and-Wait (TAW) và Just-enough-Time (JET). Ở cả hai phương pháp này, một header được gửi ra trước burst dữ liệu để cấu hình cho bộ chuyển mạch dọc trên tuyến đường của burst dữ liệu. Sau đây, chúng ta tìm hiểu về hai phương pháp báo hiệu này.

### 3.2.2 Giao thức báo hiệu JET (Just Enough Time)

Hình 3.2 minh họa cho giao thức báo hiệu JET. Như ta thấy, đầu tiên node nguồn gửi ra một gói header của burst (Burst header packet - BHP) trên kênh điều khiển về phía node đích. Gói BHP được xử lý tại mỗi node phía sau để thiết lập một đường truyền dữ liệu toàn quang cho burst dữ liệu tương ứng. Nếu việc dự trữ thành công, bộ chuyển mạch sẽ được cấu hình trước khi burst dữ liệu tới. Trong lúc đó burst dữ liệu đợi tại node nguồn trong miền điện. Sau một khoảng thời gian đã định trước offset time, burst dữ liệu được gửi toàn quang trên bước sóng đã chọn. Khoảng thời gian offset time được tính toán dựa trên số hop từ node nguồn tới node đích và thời gian chuyển mạch tại mỗi node lõi. Offset time được tính bằng công thức: OT = h. + ST, với h là số hop giữa node nguồn và node đích,  là thời gian xử lý header của burst tại mỗi hop, và ST là thời gian cấu hình cho bộ chuyển mạch. Nếu tại bất kì node trung gian nào việc dự trữ không thành công thì burst sẽ bị hủy. Điểm khác biệt của JET khi so sánh với các phương pháp báo hiệu một chiều khác là dự trữ có trì hoãn và giải tỏa không tường minh.



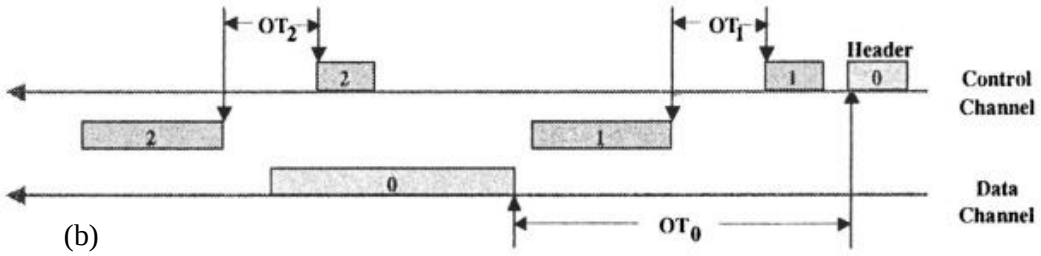

Hình 3.2: Giao thức báo hiệu JET

Thông tin về thời điểm bắt đầu và kết thúc của tất cả các burst được sắp xếp vào kênh truyền cần phải được duy trì cho mỗi kênh ở mỗi cổng xuất của từng bộ chuyển mạch cho JET, điều này làm cho hệ thống trở nên phức tạp hơn. Mặt khác, JET có thể dò tìm được vị trí mà ở đó không có xuất hiện xung đột khi truyền burst, mặt dù khởi điểm của một burst mới đến có thể sẽ sớm hơn thời điểm kết thúc của một burst đã được chấp nhận trước nó, có nghĩa là một burst có thể sẽ được truyền đi ở giữa hai burst đã dự trữ kênh truyền rồi (nếu chiều dài burst mới này thích hợp). Vì vậy burst có xác suất được chấp nhận cao hơn trong giao thức JET.

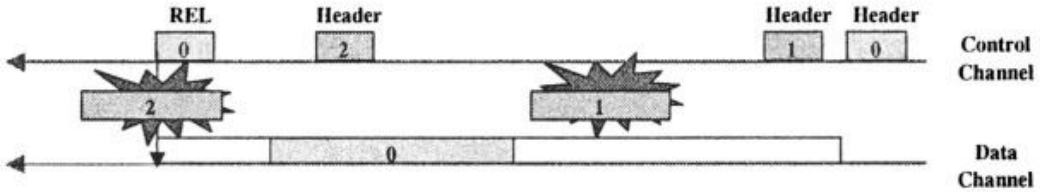
Có nhiều kiểu báo hiệu có liên quan mật thiết với kiểu báo hiệu một chiều như Tell-And-Go (TAG) và Just-In-Time (JIT). Trong phương pháp TAG, burst dữ liệu phải được làm trễ lại tại mỗi node để cho phép có thời gian xử lý header của burst giúp cấu hình cho bộ chuyển mạch thay vì chỉ định trước khoảng thời gian này tại node nguồn và thời gian hoãn này được đặt trong offset time. Để làm trễ các burst dữ liệu lại như thế, đòi hỏi dùng đến sợi quang làm trễ fiber delay lines (FDL), cấu tạo gồm nhiều vòng sợi quang. Khoảng thời gian bị trễ khi dữ liệu truyền đi bên trong FDL chính là lượng thời gian mà dữ liệu được làm trễ.

Hoạt động của JIT giống như JET nhưng khác ở chỗ JIT dùng cách dự trữ tức thời và giải tỏa tường minh thay vì dùng dự trữ có trì hoãn và giải tỏa không tường minh. Hình 3.4 (a) và (b) so sánh giữa hai phương pháp JIT và JET với cùng một kịch bản báo hiệu.

(a)




(b)



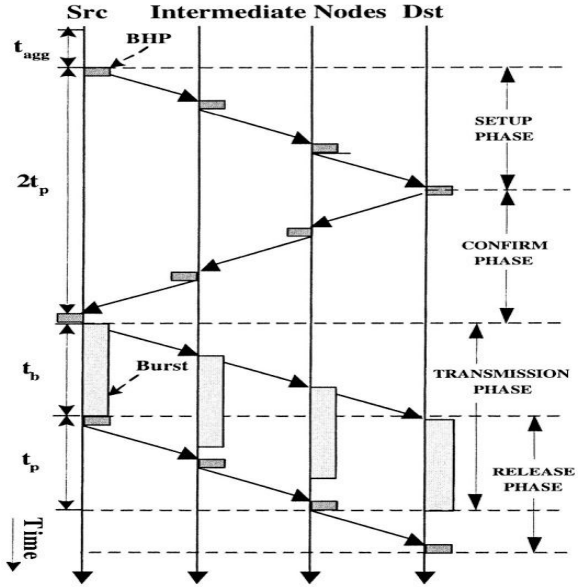

Hình 3.3: So sánh báo hiệu JET (a) và JIT (b).

Lợi ích chính của cách dùng giao thức báo hiệu một chiều là giảm thiểu thời gian trễ khi truyền dữ liệu từ đầu cuối tới đầu cuối trên mạng trục (backbone) giúp giảm khả năng mất gói do xung đột burst vì tranh giành nguồn tài nguyên trong mạng lõi không dùng bộ đệm.

### 3.2.3 Giao thức báo hiệu TAW (Tell and Wait)

Hình 3.4 minh họa phương pháp báo hiệu TAW. Với TAW, bản tin BHP thiết lập được gửi đi dọc theo tuyến đường mà burst dữ liệu đi để thu thập thông tin về kênh đang sẵn sàng tại mỗi node. Tại đích, một giải thuật cấp phát kênh được thực thi, và thời điểm dự trữ mỗi link sẽ được xác định dựa trên thời điểm sớm nhất mà một kênh ở mỗi node trung gian sẵn sàng. Một bản tin BHP xác nhận được gửi ngược trở về phía nguồn để dự trữ kênh truyền cho khoảng thời gian cần thiết tại mỗi node. Tại bất kì node nào trên đường truyền, nếu kênh cần dùng đã bị dự trữ rồi

thì một bản tin BHP giải tỏa được gửi về đích để giải tỏa hết các tài nguyên trước đã được dự trữ thành công. Còn nếu bản tin xác nhận tới được nguồn thì burst dữ liệu sẽ được gửi đi vào mạng lõi.




Hình 3.4: Giao thức báo hiệu TAW

Cũng nói thêm, TAW giống với mạng định tuyến theo bước sóng, kênh truyền có thể được dự trữ theo hướng xuôi như phương pháp dự trữ được tạo ở node nguồn (SIR) hay dự trữ theo hướng ngược lại từ phía đích trở về nguồn như ở phương pháp dự trữ được tạo ở node đích (DIR). TAW trong OBS khác với mạng định tuyến theo bước sóng WDM ở chỗ là tài nguyên của các node chỉ được dự trữ trong khoảng chiều dài của burst. Và nếu chiều dài của burst đã được biết trước trong quá trình dự trữ thì phương pháp giải tỏa không tường minh sẽ được dùng kèm theo nhằm tận dụng tối đa hiệu quả băng thông. Tất cả các giao thức mà ta đề cập đến ở trên đều là các giao thức báo hiệu một chiều ngoại trừ TAW là giao thức báo hiệu hai chiều. Nếu ta so sánh giữa TAW và JET, nhược điểm của TAW là trễ nhiều do ở thời gian thiết lập round-trip, chính là thời gian mà ta dùng để thiết lập các kênh; tuy nhiên, ở TAW việc mất burst xảy ra rất thấp. Vì vậy mà TAW rất phù hợp cho lưu lượng dễ mất: loss-sensitive traffic. Còn ở JET, thời gian trễ ít hơn vì chỉ là tổng của thời gian lan truyền theo một chiều và một offset time. Không có giao thức báo hiệu nào cho ta tính mềm dẻo giữa giá trị mất mát và thời gian trễ.

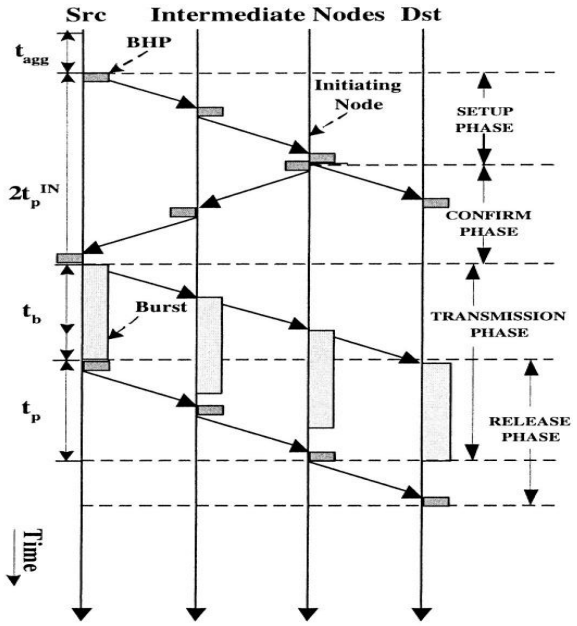
### 3.2.4 Báo hiệu được khởi tạo tại node trung gian INI (Intermediate Node Initiated)

Nhiều giao thức báo hiệu được đưa ra để áp dụng cho việc truyền dữ liệu trong mạng toàn quang OBS. Để đáp ứng cho yêu cầu dự trữ tài nguyên động cho việc truyền các burst dữ liệu, đầu tiên phương pháp báo hiệu phải tìm ra tuyến đường thích hợp từ nguồn tới đích, sau đó mới sắp xếp dữ liệu vào một kênh bước sóng riêng nào đó tại mỗi node trung gian. Giao thức báo hiệu phân bố phổ biến nhất đã được nghiên cứu là Tell-And-Wait (TAW) và just-enough-time (JET). TAW là báo hiệu hai chiều dựa trên thông tin hồi đáp, dùng các bản tin điều khiển thiết lập và giải tỏa tường minh. JET là giao thức báo hiệu một chiều không cần thông tin hồi đáp, dùng các gói header của burst – BHP (burst header packet) có tính ước lượng để giải tỏa và thiết lập. Để khỏi phải chuyển đổi quang điện trong lõi, các phương pháp báo hiệu có một offset time giữa BHP và dữ liệu tương ứng của nó. Trong BHP có chứa thông tin về chiều dài của burst, kết hợp với thông tin về offset time, báo cho node biết được thời điểm node này cần cấu hình bộ chuyển mạch cho burst dữ liệu sắp tới. Khoảng thời gian offset time cho phép BHP được xử lý tại node trung gian trước khi burst dữ liệu tới node trung gian đó. Nếu ta đem so sánh giữa TAW và JET, nhược điểm của TAW là trễ do round-trip time, nhưng bù lại rất ít mất dữ liệu, vì vậy TAW phù hợp cho loss-sensitive traffic. Về phía JET, mất dữ liệu dễ xảy ra, nhưng độ trễ khi truyền dữ liệu từ đầu cuối này tới đầu cuối khác ít hơn TAW. Trong TAW phải mất 3 lần độ trễ lan truyền từ nguồn tới đích thì burst mới tới được đích, trong khi đó JET chỉ cần lần trễ lan truyền một chiều và một khoảng offset time. Như ta đã nói, chưa có phương pháp báo hiệu riêng biệt nào cho phép kết hợp uyển chuyển giữa độ trễ và việc mất dữ liệu. Trong mạng IP over OBS, người ta mong muốn cung cấp hỗ trợ chất lượng dịch vụ cho các ứng dụng đòi hỏi nhiều yêu cầu về chất lượng dịch vụ khác nhau, chẳng hạn như voice-over-IP, video-on-demand, hay video conferencing. Nhiều giải pháp được đưa ra để hỗ trợ chất lượng dịch vụ trong mạng lõi OBS. Tuy nhiên, không có phương pháp đơn lẻ (không có sự kết hợp giữa các giao thức lại) nào cho phép hỗ trợ một cách

mềm dẻo cả hai yêu cầu về độ trễ và mất dữ liệu trong mạng OBS. Có một số phương pháp cải thiện QoS, ví dụ như JET kết hợp với offset time dành cho các lớp lưu lượng khác nhau, chịu được xác suất nghẽn cao. Trong phương pháp này, node nguồn phải ước lượng trước offset time để có thể hỗ trợ cho các yêu cầu khác nhau của các lớp gói dữ liệu.

Để khắc phục các hạn chế của hai phương pháp TAW và JET, phương pháp báo hiệu được khởi tạo ở node trung gian INI được đưa ra. Trong phương pháp INI, một node ở giữa node nguồn và node đích nằm trên đường truyền được chọn làm node khởi tạo (initiating node). Tại node khởi tạo này, một thuật toán dự trữ kênh sẽ được thực hiện nhằm xác định thời gian sớm nhất mà burst có thể được gửi đi ở node nguồn và thời gian sớm nhất tương ứng mà tại đó các node ở giữa node nguồn với node khởi tạo có thể được sắp xếp để nhận burst dữ liệu tới. Việc dự trữ thật sự các kênh ở node khởi tạo bắt đầu theo cả hai hướng: từ node khởi tạo tới node nguồn lẫn từ node khởi tạo về node đích. Việc lựa chọn node khởi tạo được chỉ ra trong giao thức báo hiệu INI. Hình 3.5s minh họa phương pháp báo hiệu INI. Khi một burst dữ liệu được hình thành tại node biên, một bản tin BHP thiết lập (setup BHP) được gửi tới node. BHP sẽ thu thập thông tin chi tiết về các kênh tại mỗi node nó đi qua cho tới khi đến node khởi tạo (initiating node). Tại node khởi tạo, thuật toán cấp phát kênh được thực thi để xác định khoảng thời gian mà kênh cần được dự trữ tại mỗi hop trung gian nằm giữa node nguồn và node khởi tạo. Kế đó, một gói xác nhận (confirm packet) được gửi ngược về node nguồn, gói này tiến hành dự trữ các kênh dọc theo đường đi của nó từ node khởi tạo tới node nguồn. Nếu có kênh bận ở bất kỳ node nào, gói giải tỏa (release packet) sẽ được gửi trở về node khởi tạo để giải tỏa hết cho các tài nguyên trước đó đã dự trữ thành công. Nếu gói xác nhận tới được nguồn thành công thì burst dữ liệu sẽ được gửi đi tại thời điểm đã được sắp xếp trước. Cùng với lúc gửi đi gói xác nhận về node nguồn, node khởi tạo cũng gửi đi một bản tin BHP thiết lập không cần trả lời (unacknowledged setup BHP) về phía node đích nhằm dự trữ trước các kênh truyền giữa node khởi

tạo và node đích. Nếu tại bất kì node nào giữa node khởi tạo và node đích mà bản tin BHP không dự trữ được kênh truyền thì burst dữ liệu sẽ bị drop ở node đó.



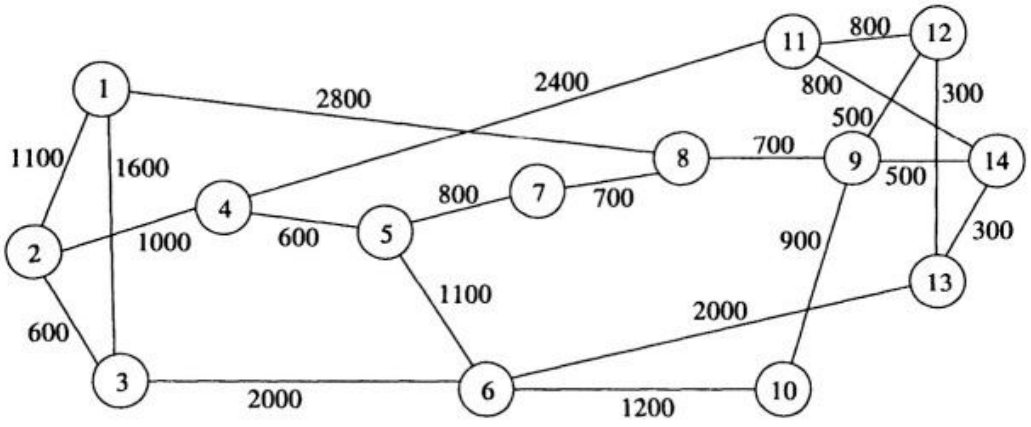

Hình 3.5: Báo hiệu được khởi tạo ở node trung gian INI.

Trong giao thức TAW, bản tin ACK được gửi từ phía đích trước khi burst dữ liệu được gửi đi từ nguồn, còn trong JET, không có ack. Ở INI, có ack xuất phát từ node khởi tạo, vì vậy giảm được xác suất nghẽn so với JET. Không những thế, vì khoảng thời gian mà burst dữ liệu phải đợi tại nguồn ít hơn khoảng thời gian trễ do lan truyền từ nguồn tới đích nên INI giảm được trễ truyền từ đầu cuối đến đầu cuối khi so sánh với TAW. Trong giao thức báo hiệu INI, nếu node khởi tạo là node nguồn thì nó trở thành báo hiệu JET, nếu node khởi tạo là node đích thì trở thành báo hiệu TAW. Trong INI, ta có thể dùng cả hai phương pháp dự trữ thông thường hay dự trữ có trì hoãn đều được. Với các dự trữ có trì hoãn thì hoạt động của giao thức báo hiệu được cải thiện hơn.

**3.2.5 Ví dụ minh họa:**

Xem đường đi 2-4-5-7 trong hình 3.6 có node 2 là node nguồn, node 7 là node đích. Ta có 4 node có thể làm node khởi tạo, bao gồm luôn cả node nguồn và node đích. Nếu ta chọn node nguồn (chính là node 2) làm node khởi tạo thì báo hiệu

INI trở thành báo hiệu JET. Nếu chọn node đích làm node khởi tạo (node 7) thì trở thành báo hiệu TAW. Các node có khả năng làm node khởi tạo khác là node 4 và node 5. Ta xét node 5 là node khởi tạo. Hoạt động của INI như sau: node 2 gửi bản tin BHP cho hop kế tiếp là node 4, có kèm theo thông tin về kênh sẵn sàng trên link 2-4. Tại node 4 thêm vào thông tin về kênh sẵn sàng trên link 4-5 sau đó gửi đi bản tin BHP tới node kế là node 5. Khi node 5 là node khởi tạo nhận được bản tin BHP, nó thực hiện một thuật toán dự trữ kênh để xác định thời gian sớm nhất mà lúc đó burst yêu cầu có thể được phục vụ bởi các node trung gian nằm giữa node nguồn với node khởi tạo, bao gồm cả node nguồn và node khởi tạo. Một gói trả lời, sẽ dự trữ kênh truyền tại các node trung gian này vào thời điểm đã được định trước, được gửi ngược về từ node khởi tạo tới node nguồn. Ngay khi gói trả lời tới được node nguồn 2 thì burst dữ liệu sẽ được gửi đi. Có một bản tin BHP được gửi từ node khởi tạo (node 5) đến node đích (node 7) và cấu hình cho node 7 chuẩn bị nhận burst dữ liệu tới vào thời điểm thích hợp. Node 7 không gửi bản tin ack về cho node khởi tạo. Bản tin BHP được gửi đi từ node khởi tạo chỉ có nhiệm vụ dự trữ kênh truyền sẵn sàng và tiếp tục đi theo hướng từ node khởi tạo về phía đích.




Hình 3.6: Cấu hình mạng 14 node.

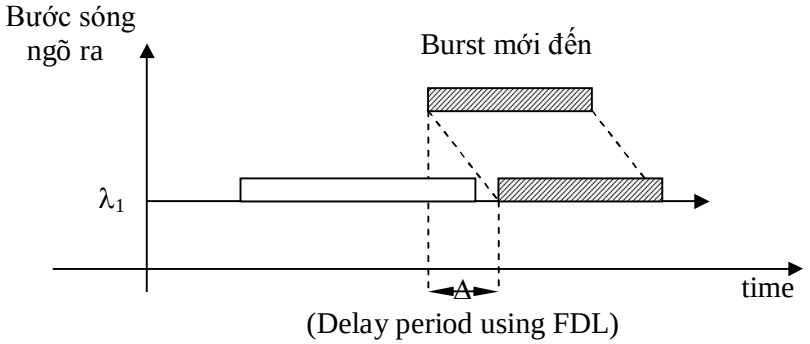
### 3.3. Các phương pháp giải quyết xung đột trong mạng OBS

Trong mạng OBS các burst được truyền từ node nguồn đến node đích sau khi được chuyển mạch qua hết các node trung gian mà không cần bộ đệm quang nên khả năng xảy ra xung đột giữa các burst là rất lớn. Xung đột có thể xảy ra khi

nhiều burst muốn rời node lõi trên cùng một tuyến WDM hay burst ở các ngõ vào khác nhau muốn đến một ngõ ra tại cùng một thời điểm. Các phương pháp giải quyết xung đột được đề xuất như sau

### 3.3.1. Các đường dây trễ quang FDL (Fiber Delay Line)

Nếu như trong miền điện tử có các bộ nhớ truy cập ngẫu nhiên như RAM thì trong miền quang ý tưởng bộ đệm quang vẫn chưa thực hiện được. Vì vậy để đệm burst dữ liệu trong một khoảng thời gian người ta chỉ có thể dùng đến các đường dây trễ quang FDL. Các burst dữ liệu được lưu giữ trong miền quang một khoảng thời gian cố định. Bằng cách kết nối các dây trễ FDL theo tầng hay kết nối song song, bộ đệm được đề xuất này có thể giữ các burst dữ liệu trong các thời gian khác nhau. Với phương pháp này, burst đang tham gia tranh chấp sẽ được làm trễ lại cho tới khi nghẽn được giải quyết. Phương pháp này dựa trên ý tưởng là: khi một bước sóng được yêu cầu lại chưa sẵn sàng thì burst dữ liệu sẽ được làm trễ lại trong một FDL cho tới khi kênh bước sóng đó trở về trạng thái sẵn sàng.




Hình 3.7: Giải quyết xung đột bằng phương pháp sử dụng đường dây trễ FDL

Ở hình trên kênh bước sóng mong muốn của burst dữ liệu là λ1 nhưng kênh này đã bị chiếm tại thời điểm tới của burst. Trong trường hợp này, burst dữ liệu sẽ được đệm lại trong khoảng thời gian Δ, khi đó kênh này đã trở về trạng thái sẵn sàng tại thời điểm tới của burst dữ liệu sau khi đã được đệm.

Do FDL dựa trên trễ truyền của cáp quang và sự truy cập liên tục nên nó có nhiều hạn chế so với RAM. Nếu dung lượng bộ đệm lớn thì số lượng và chiều dài

của FDL càng tăng nên dễ gây tổn hao và việc sử dụng bộ đệm cũng không thể hoàn toàn giảm khả năng mất burst

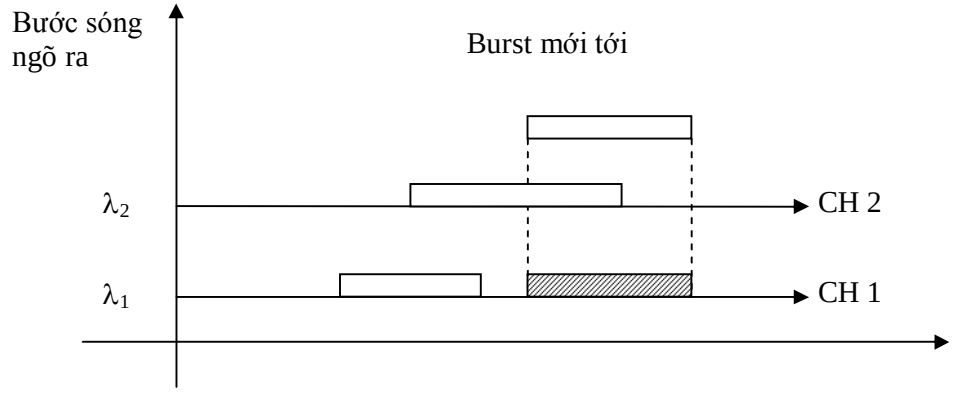
### 3.3.2. Bộ chuyển đổi bước sóng

Sử dụng bộ chuyển đổi bước sóng wavelength converter để chuyển đổi kênh ngõ ra khác cho burst dữ liệu nếu như kênh nó mong muốn đã bị chiếm giữ tại thời điểm burst tới node. Trong WDM, nhiều bước sóng được ghép cùng một lúc trên một liên kết nối hai chuyển mạch chùm quang. Nhiều bước sóng có thể giảm tối đa số lượng xung đột. Giả sử có hai burst cùng đi đến một đích và ra ở cùng ngõ ra tại một thời điểm. Cả hai burst vẫn có thể truyền đi tiếp nếu ở trên hai bước sóng khác nhau.

Chuyển đổi bước sóng là quá trình chuyển đổi một bước sóng ở ngõ vào thành một bước sóng khác ở ngõ ra, do vậy làm tăng khả năng sử dụng lại bước sóng nghĩa là tất cả các kênh bước sóng trên cùng một cáp quang có thể được dùng chung bởi tất cả các burst.

Có các kiểu chuyển đổi sau:

- Chuyển đổi toàn bộ (Full conversion): Một bước sóng có thể chuyển thành bất kì bước sóng nào ở đầu ra, do vậy không có một bước sóng nào xuất hiện liên tục trên một kết nối từ đầu cuối đến đầu cuối.
- Chuyển đổi có giới hạn (Limitted conversion): Việc chuyển đổi bước sóng bị giới hạn để không phải tất cả các kênh ngõ vào đều có thể kết nối đến kênh ngõ ra. Việc giới hạn này sẽ làm giảm chi phí của chuyển mạch trong khi chấp nhận một số lượng xung đột
- Chuyển đổi cố định (Fixed conversion): Đây cũng là một dạng của chuyển đổi có giới hạn, trong đó một kênh ngõ vào được kết nối với một hay nhiều kênh ngõ ra được chỉ định trước
- Chuyển đổi một phần (Sparse conversion): Trong mạng có thể bao gồm các node có chuyển đổi toàn bộ có giới hạn, cố định và không có bộ chuyển đổi bước sóng




Hình 3.8: Giải quyết xung đột bằng phương pháp chuyển đổi bước sóng.

### 3.3.3. Định tuyến chuyển hướng

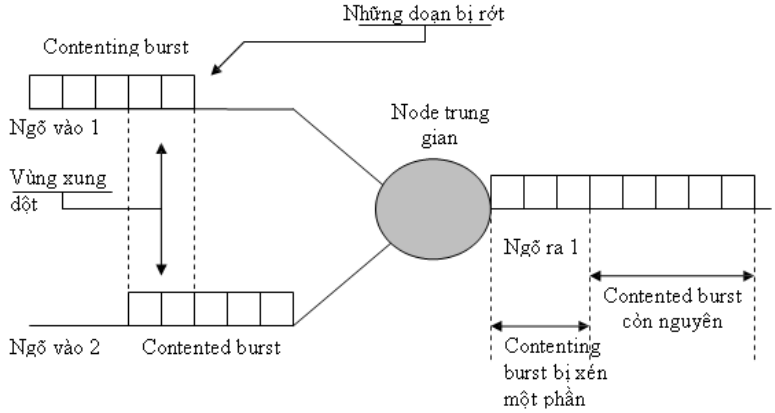
Trong định tuyến chuyển hướng, xung đột được giải quyết bằng cách định tuyến burst dữ liệu đến một ngõ ra khác thay vì ngõ ra ban đầu, tức là kể từ node đó đi theo con đường khác để đến đích chứ không còn đi theo con đường ngắn nhất ban đầu. Định tuyến chuyển hướng không được quan tâm đối với mạng chuyển mạch gói trong miền điện, tuy nhiên nó lại thực sự cần thiết trong mạng toàn quang khi chưa có bộ đệm quang. Trong định tuyến chuyển hướng, gói hay burst dữ liệu bị chuyển hướng có thể đi trên con đường dài hơn để đến đích làm tăng độ trễ và giảm chất lượng tín hiệu. Hơn nữa, có thể một gói sẽ bị vòng lặp (loop) trong mạng do không tìm được đường đến đích hay bị chuyển hướng quá nhiều và thêm tắc nghẽn trong mạng.

Một số vấn đề khác trong định tuyến chuyển hướng là bảo trì thời gian offset giữa gói điều khiển và gói dữ liệu của một burst bị chuyển hướng. Bởi vì burst bi chuyển hướng phải đi trên đường có số node trung gian nhiều hơn khi nó không bị chuyển hướng, do đó thời gian offset trước đây là không đủ để các chuyển mạch kế tiếp xử lý các gói điều khiển trước khi burst dữ liệu đến. Để khắc phục vấn đề này, nhiều xử lý được thêm vào để tính toán lại thời gian offset. Một cách đơn giản hơn là chỉ cần loại bỏ những burst có thời gian offset không hợp lệ. Để biết được số node trung gian mà burst phải đi qua ta có thể dùng các bộ đếm.

### 3.3.4. Phân đoạn burst

Trong phương pháp này burst được phân thành các đoạn để khi có xung đột thì chỉ có một phần burst bị mất, phần còn lại vẫn được truyền qua mạng. Phần burst bị mất có thể là phần trước hay phần sau. Nếu burst bị hủy bỏ phần đầu thì phần burst còn lại cần thông tin của offset giữa gói tin điều khiển của burst và điểm đầu của phần burst còn lại. Nếu burst bị hủy bỏ phần đuôi thì cần phải thêm gói tin điều khiển của burst cho phần đuôi bị hủy bỏ.

Nếu như ranh giới giữa các đoạn  hoàn toàn trong suốt trong mạng lõi toàn quang thì các node biên phải chịu trách nhiệm định nghĩa và xử lý các đoạn trong miền




Hình 3.9: Giải quyết xung đột bằng phương pháp phân đoạn burst

điện. Hơn nữa, node nhận phải có khả năng nhận ra điểm bắt đầu của mỗi đoạn và xác định xem thử đoạn đó còn nguyên vẹn hay không, do đó một số header dùng để nhận ra lỗi và sửa lỗi chứa trong một đoạn. Thêm vào đó thông tin về tín hiệu đồng hồ có thể cũng cần phải có trong mỗi header của mỗi đoạn để node nhận ngõ ra có thể xác định và phục hồi dữ liệu trên mỗi đoạn. Khi các đoạn có chiều dài không đổi thì việc đồng bộ ở máy thu trở nên dễ dàng, tuy nhiên những đoạn có chiều dài thay đổi lại có khả năng chứa được những gói có chiều dài khác nhau. Kích thước của mỗi đoạn còn phải cân nhắc giữa mất mát trong một lần xung đột và số lượng header trong một burst. Đoạn dài sẽ dẫn đến mất nhiều dữ liệu cho mỗi lần xung đột, tuy nhiên những đoạn dài cũng dẫn đến overhead và tỉ số giữa chiều dài header sovới chiều dài payload sẽ nhỏ theo. Một số vấn đề khác trong phân đoạn burst là

quyết định xem đoạn nào bị rớt khi xung đột xảy ra giữa hai burst. Giả sử gọi burst bị xung đột là contented burst còn burst xung đột là contenting burst. Chú ý rằng burst được xem là contented hay contenting burst phụ thuộc vào thứ tự của nó đến chuyển mạch chứ không phải thứ tự của gói điều khiển đến trước hay đến sau. Có hai cách để xác định xem những đoạn nào nên rớt, được gọi là tail-dropping (rớt phần đuôi) và head-dropping (rớt phần đầu).

Trong cách rớt phần đuôi tail-dropping thì các đoạn chồng lấn của contented burst sẽ bị rớt còn trong cách rớt phần đầu heading-dropping thì các đoạn của contenting burst chồng lấn sẽ bị rớt. Ưu điểm của việc tail-dropping so với tail-dropping trong việc thay đổi các gói sai thứ tự ở node đích với giả thuyết rằng các gói rớt được truyền lại sau đó. Việc head-dropping làm cho các gói đến đích sai thứ tự, tuy nhiên, ưu điểm của head-dropping là nó chắc chắn rằng một khi burst đến một node không bắt gặp một xung đột nào và sao đó các burst này tiếp tục đi đến đích mà không phụ thuộc vào các burst đi sau nó có mức ưu tiên nào đi chăng nữa.

**3.4 Kết luận chương**

Vậy là trong chương này em đã trình bày các giao thức báo hiệu cũng như các phương pháp giải quyết xung đột trong mạng OBS. Có nhiều các kỹ thuật dự trữ và giải phóng tài nguyên nhưng kỹ thuật báo hiệu một chiều JET với dự trữ có trì hoãn và giải tỏa không tường minh cho xác suất burst được chấp nhận cao hơn các kỹ thuật khác được đề nghị sử dụng trong mạng OBS. Việc lựa chon giữa các biện pháp giải quyết xung đột cũng là một vấn đề quan trong nhằm giảm tỉ lệ mất burst đến mức thấp nhất có thể. Tùy theo yêu cầu cụ thể của từng mạng và điều kiện cho phép mà ta chọn ra phương pháp thích hợp hay để tận dụng các ưu điểm của mỗi phương pháp ta có thể sử dụng kết hợp chúng sẽ cho hiệu quả giảm tỉ lệ mất burst cao hơn nhiều so với việc dùng riêng lẽ từng phương pháp.

**Chương 4**

# CÁC GIẢI THUẬT XẾP LỊCH TRONG MẠNG OBS

## 4.1 Giới thiệu chương

Khi một burst tới một node, nó cần được cấp cho một kênh bước sóng ở ngõ ra, vì vậy tất cả các node trong hệ thống mạng đều phải có bộ wavelength converter. Ngoài ra, nhằm làm giảm thiểu khoảng thời gian trống giữa 2 burst truyền đi trên cùng một kênh bước sóng, người ta dùng thêm bộ sắp xếp các burst tại tất cả các node tham gia trong mạng, bộ đó được gọi là bộ xếp lịch (channel scheduling).

Khi header của burst dữ liệu tới được nút lõi, các thông số về burst dữ liệu sẽ được nhận biết ở nút lõi như chiều dài burst (burst duration), thời gian burst đó tới nút (arrival time)… Dựa vào những thông số này, nút lõi sẽ xác định được kênh bước sóng thích hợp nhất dành cho burst dữ liệu nhờ thuật toán sắp xếp của bộ channel scheduling.

Thuật toán sắp xếp kênh bước sóng cho các kênh dữ liệu được chia làm hai phần chính như sau: có hoặc không có sử dụng void filling (lấp đầy khoảng trống). Trong phần này, em sẽ trình bày về 2 loại xếp lịch này dựa trên hai giải thuật cơ bản là FFUC (First Fit Unscheduled Channel) và LAUC (Latest Available Unscheduled Channel).

## 4.2 Các thông số sử dụng trong các thuật toán sắp xếp

Các thông số được sử dụng cho hầu hết các loại thuật toán sắp xếp là:

$L_b$: Chiều dài burst chưa được sắp xếp.

$t_{ub}$: Thời gian tới của burst chưa được sắp xếp.

$W$: Số kênh dữ liệu ngõ ra.

$N_b$: Số burst tối đa dùng trên một kênh ngõ ra.

$D_i$: Kênh ngõ ra thứ i.

$LAUT_i$: Thời gian rỗi sớm nhất của kênh thứ i, dùng cho bộ xếp lịch ko sử dụng void filling.

$S_{(i,j)}$, $E_{(i,j)}$: Thời điểm bắt đầu và kết thúc của mỗi burst thứ j đã được sắp xếp trên kênh thứ i.

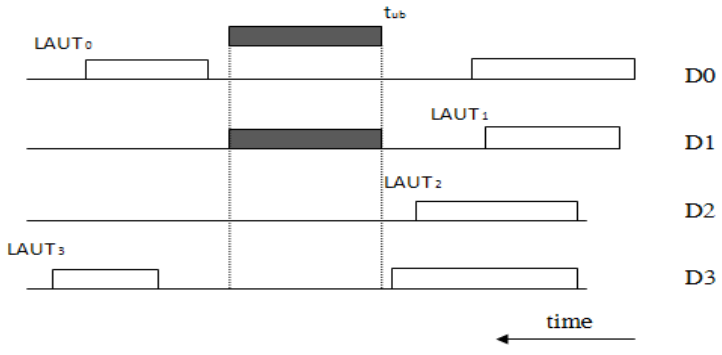
$Gap_i$: Nếu kênh rỗi, gap là sự chênh lệch giữa thời gian đến của burst và các thông số $LAUT_i$ đối với trường hợp không sử dụng void filling, và thông số $E_{(i,j)}$ đối với trường hợp có void filling. Thông số Gap là cơ sở để thuật toán quyết định nên sử dụng kênh nào khi có hơn 1 kênh rỗi. Trong trường hợp kênh không rỗi, hệ số gap bằng 0.

## 4.3 Các giải thuật xếp lịch cơ bản

### 4.3.1 Các thuật toán không sử dụng void-filling

#### 4.3.1.1 Thuật toán FFUC

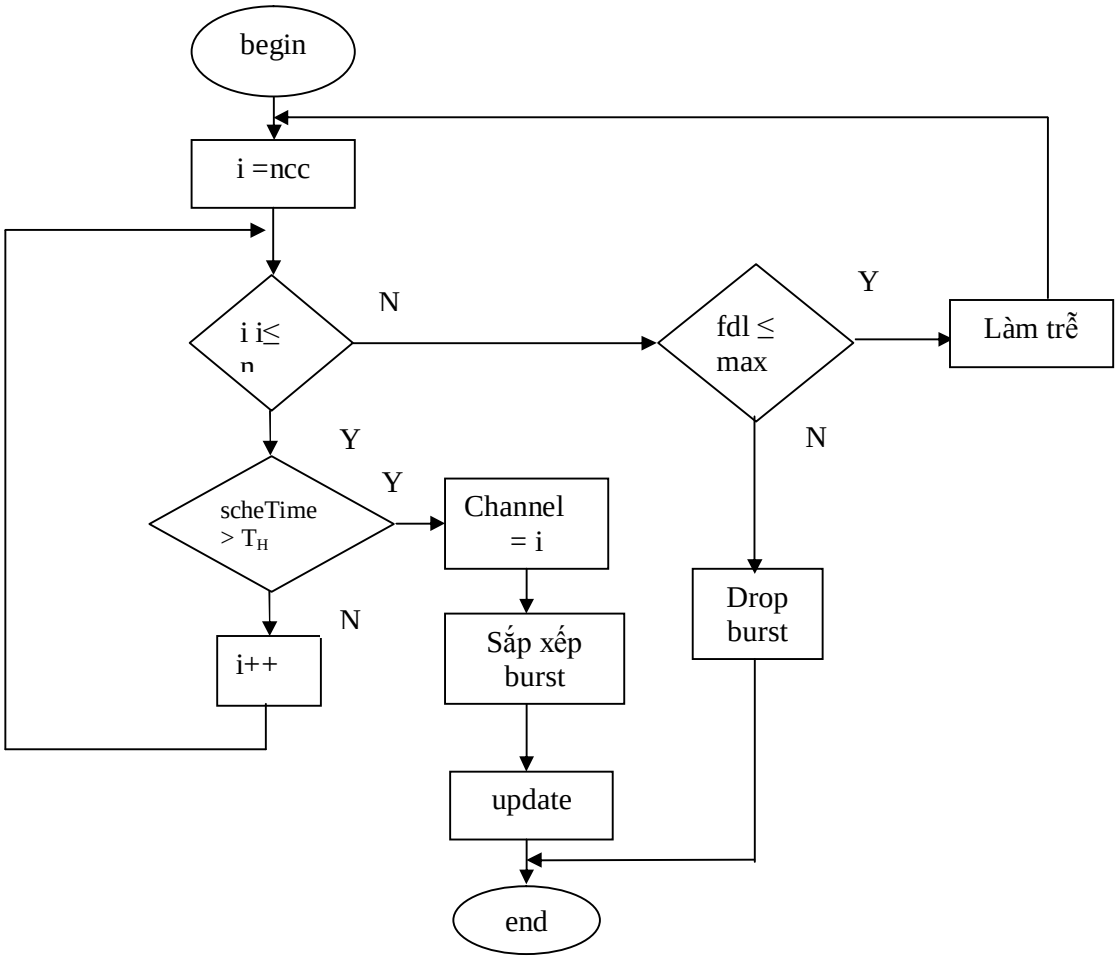
Giải thuật FFUC (First Fit Unschedule Channel) không sử dụng void filling có thể được trình bày cơ bản như sau: Khi một burst dữ liệu đến một nút. Nút đó sẽ so sánh thông số $Gap_i = t_{ub} - LAUT_i$, nếu thông số này lớn hơn 0 thì kênh đó sẽ thích hợp để cấp cho burst đó. Trong trường hợp có nhiều hơn 1 kênh thích hợp, thuật toán sẽ chọn kênh có hệ số i thấp nhất.




Hình 4.1: Mô hình giải thuật FFUC không sử dụng void filling.

Trong ví dụ trên, ta thấy khi burst dữ liệu đến nút lõi thì có 2 kênh không thỏa mãn yêu cầu của thuật toán là kênh 0 và kênh 3 do hệ số LAUT lớn, trong khi

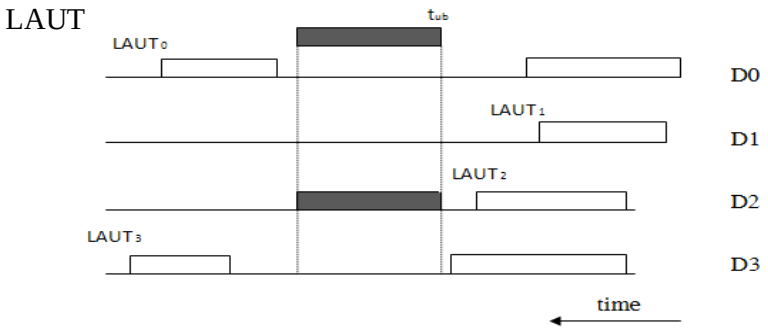
đó kênh 1 và kênh 2 là 2 kênh thỏa điều kiện của thuật toán. Trong trường hợp này, thuật toán sẽ chọn lựa kênh 1 (1<2) để là kênh ngõ ra cho burst dữ liệu.




Hình 4.2: Lưu đồ giải thuật FFUC

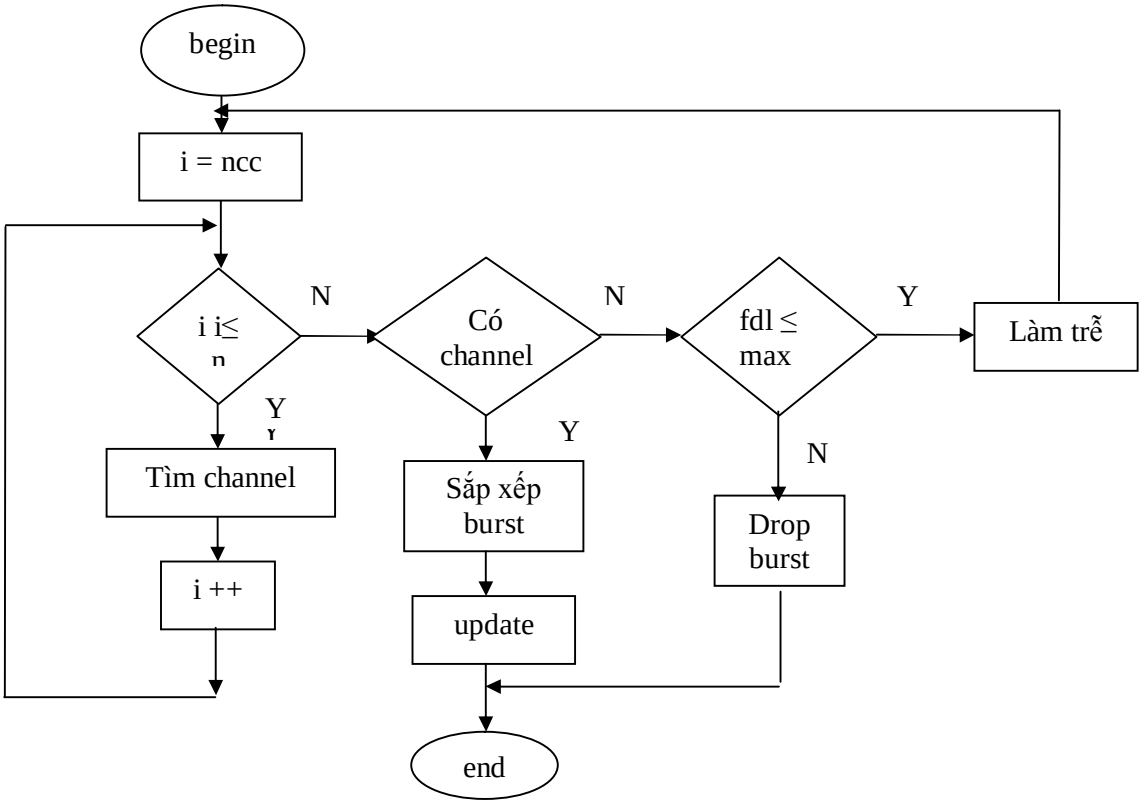
**4.3.1.2 Giải thuật LAUC**

Giải thuật LAUC (Latest Available Unschedule Channel) không sử dụng void filling có thể được trình bày cơ bản như sau: Khi một burst dữ liệu đến một nút. Nút đó sẽ so sánh thông số $Gap_i = t_{ub} - LAUT_i$, nếu thông số này lớn hơn 0 thì kênh đó sẽ thích hợp để cấp cho burst đó. Trong trường hợp có nhiều hơn 1 kênh thích hợp, thuật toán sẽ chọn kênh có hệ số gap nhỏ nhất.




Hình 4.3: Mô hình giải thuật LAUC không sử dụng void filling.

Trong trường hợp trên, cũng chỉ có 2 kênh thỏa mãn yêu cầu của thuật toán, nhưng thuật toán sẽ chọn kênh thứ 2 do có hệ số gap nhỏ hơn kênh thứ 1.




Hình 4.4: Lưu đồ giải thuật LAUC

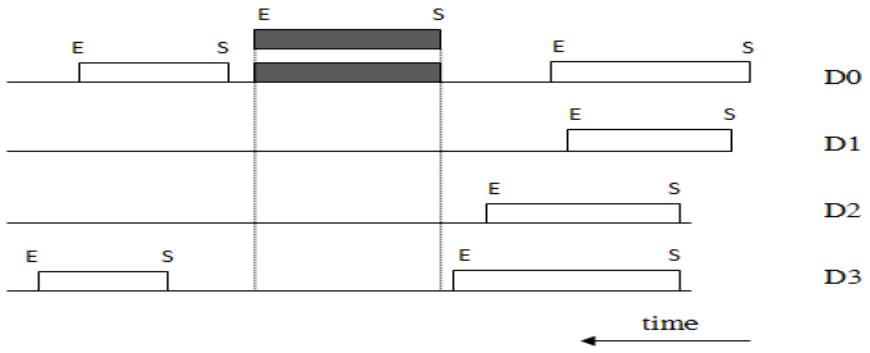
### 4.3.2 Giải thuật có sử dụng void filling

Giải thuật FFUC và LAUC có mức sử dụng tài nguyên thấp do nó không quan tâm đến các khoảng trống do đó người ta đưa ra một thuật toán khác sửa đổi từ giải thuật FFUC và LAUC ban đầu gọi là FFUC có sử dụng void filling (FFUC-VF) và giải

thuật LAUC có sử dụng void filling (LAUC-VF). Trong các thuật toán có sử dụng void filling, khoảng trống giữa các burst và khoảng trống tính từ thời điểm sử dụng sau cùng của kênh dữ liệu hay thời gian kết thúc cuối cùng của burst cuối cùng được sắp xếp trên kênh dữ liệu đến vô cùng được tận dụng để sắp xếp các burst.

#### 4.3.2.1 Giải thuật FFUC_VF

Các giải thuật sử dụng void filling thì bộ channel scheduling sẽ phải ghi nhận thông số bắt đầu và kết thúc của từng burst dữ liệu trên kênh truyền. Khi một burst dữ liệu đến, nếu thời điểm bắt đầu burst dữ liệu lớn hơn thời điểm kết thúc của burst trước đó và thời điểm kết thúc của burst dữ liệu nhỏ hơn thời điểm bắt đầu của burst liền sau nó (nếu sau nó không còn burst nào khác thì thời gian bắt đầu đó xem như là ∞) thì kênh truyền đó được chọn làm ngõ ra cho burst dữ liệu. Tương tự như trên, FFUC sẽ chọn kênh có hệ số i nhỏ nhất làm kênh ngõ ra.

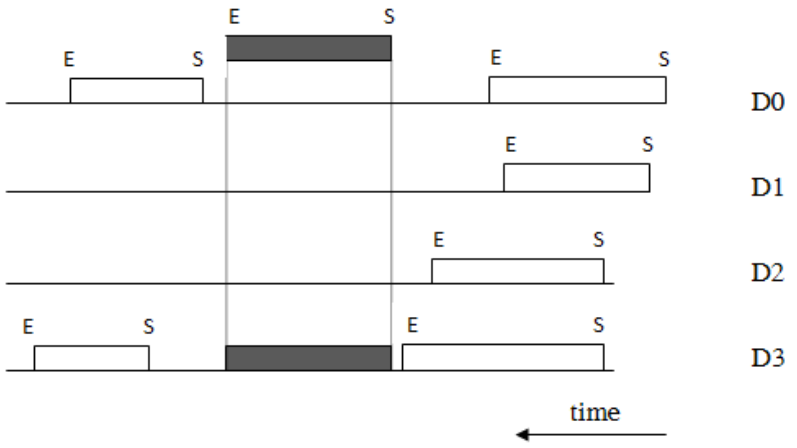



Hình 4.5 Mô hình giải thuật FFUC có sử dụng void filling.

Trong trường hợp trên thì cả 4 kênh đều thỏa mãn điều kiện của thuật toán, nhưng thuật toán FFUC sẽ chọn kênh đầu tiên (kênh 0) làm kênh ngõ ra cho burst dữ liệu.

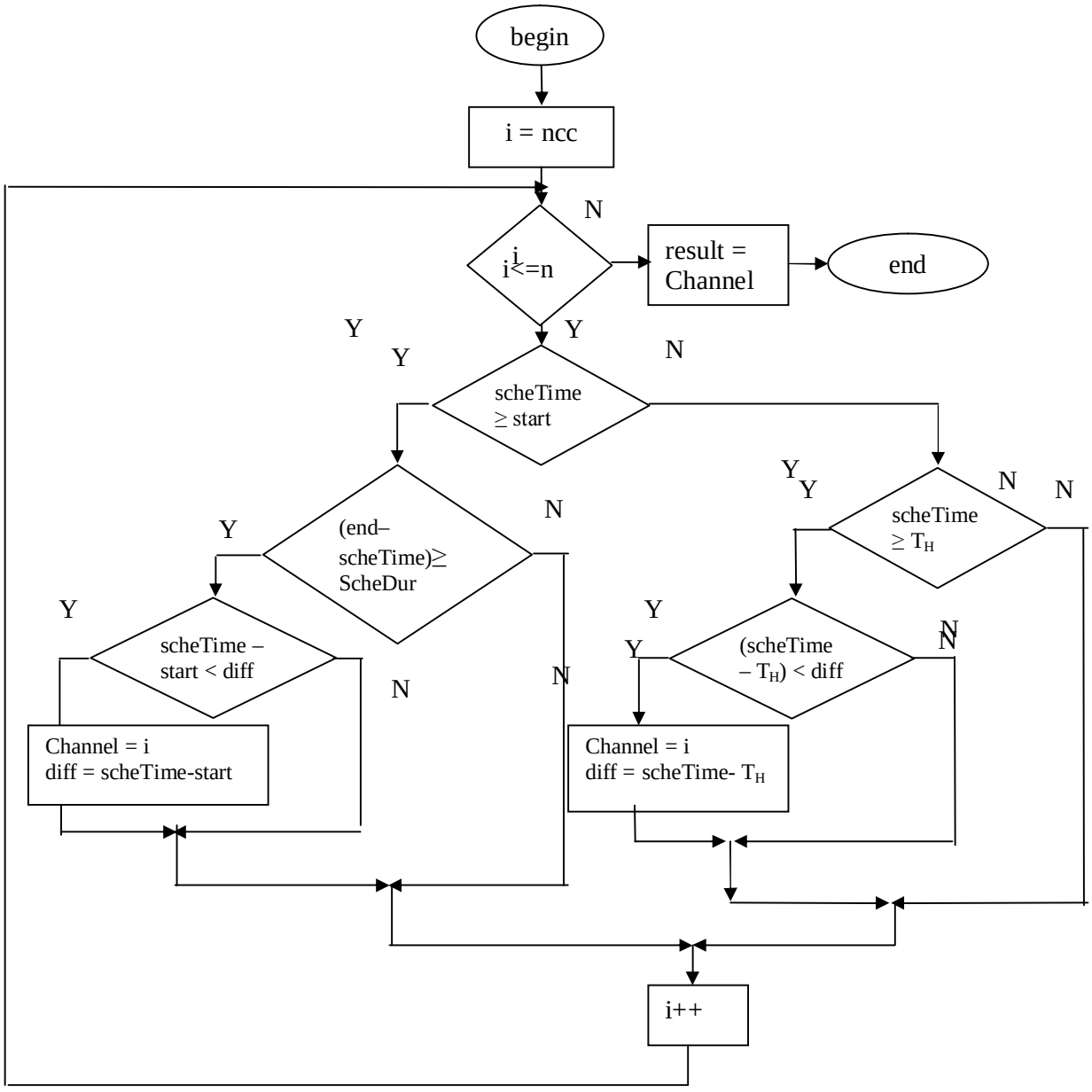
#### 4.3.2.2 Thuật toán LAUC_VF

Cũng tương tự như thuật toán FFUC_VF, thuật toán LAUC có sử dụng void filling cũng thực hiện việc xác định kênh ngõ ra cho burst dữ liệu dựa vào các thông số là thời điểm bắt đầu và kết thúc của từng burst dữ liệu được truyền trên kênh truyền. Nhưng chỉ khác ở chỗ nếu có nhiều hơn 1 kênh đủ điều kiện, thì LAUC sẽ chọn kênh rỗi gần nhất thay vì là kênh rỗi đầu tiên.




Hình 4.6 : Mô hình thuật toán LAUC có sử dụng void filling.

Trong trường hợp này cả 4 kênh đều đủ điều kiện nhưng LAUC sẽ chọn kênh số 3 do có thời gian rỗi gần với burst dữ liệu nhất.




Hình 4.7: Lưu đồ giải thuật LAUC_VF

Tóm lại:

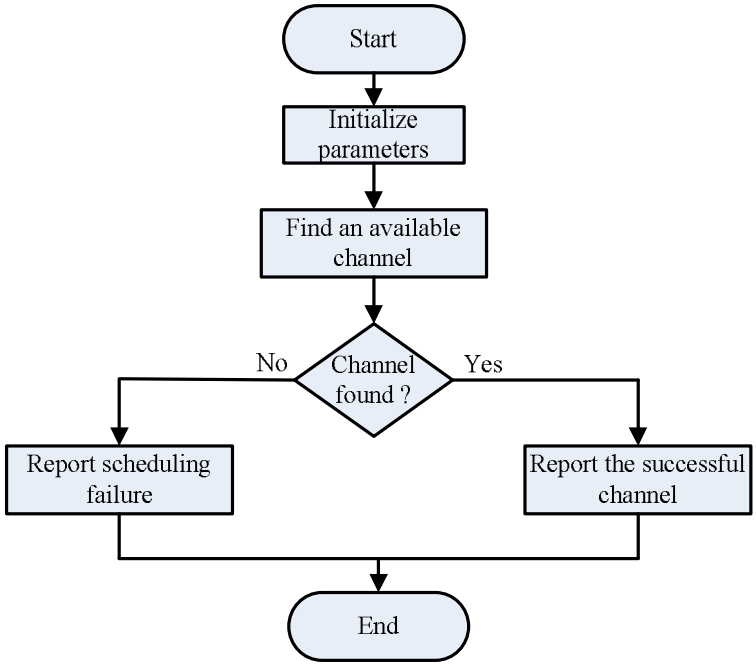
- Thuật toán FFUC là một thuật toán khá đơn giản, dễ thực hiện, nhưng bù lại khả năng mất burst dữ liệu của thuật toán này khá cao.
- Thuật toán LAUC hay còn gọi là horizon phức tạp hơn, nhưng nó lại cho hiệu quả cao hơn so với FFUC.

- Việc sử dụng void filling sẽ làm tăng hiệu quả kênh truyền dữ liệu hơn, đồng thời nó cũng làm giảm tỉ lệ mất burst đáng kể cho hệ thống.
- Chất lượng hệ thống sẽ cải thiện rất nhiều nếu sử dụng chung với FDL (Fiber Delay Line).

### 4.3.3 Vấn đề sử dụng các đường dây trễ quang FDL trong các giải thuật xếp lịch

Để giảm tỉ lệ mất burst ta có thể sử dụng các đường dây trễ quang FDL. Các tính chất cũng như hoạt động của FDL đã được trình bày trong phần các phương pháp giải quyết xung đột ở chương 2

#### 4.3.3.1 Thuật toán không sử dụng FDL




Hình 4.8 : Lưu đồ thuật toán không sử dụng FDL

Totalchannel: Số kênh sử dụng trong mạng.

Ncc: Số kênh dành cho burst header

Time gap: Tham số xem xét xem coi có sắp xếp được burst dữ liệu vào kênh truyền hay không .

startTime: thời điểm tới của burst dữ liệu.

horizon_[i]: Thời điểm rỗi của kênh thứ i.

- ***Thuật toán FirstFit:***

```
 unsigned int ndc = totalchannel_ - ncc_; // số kênh dữ liệu
 int ch = UNAVAILABLE;
   for( int i = 0; i < ndc; i++ ) {
        double time_gap = startTime - horizon_[i];//  horrizon_[i]:
        if ( time_gap >= 0.0 )
{
                    ch = i;
                    break;
        } // end of >= 0.0
  } // end of for
        if ( ch != UNAVAILABLE ) {
        result.Fflag() = FOUND;
     result.LambdaID() = ch;
     result.StartTime() = startTime;
 } else {
        result.Fflag() = NOT_FOUND;
 }
   return (result);
}
```

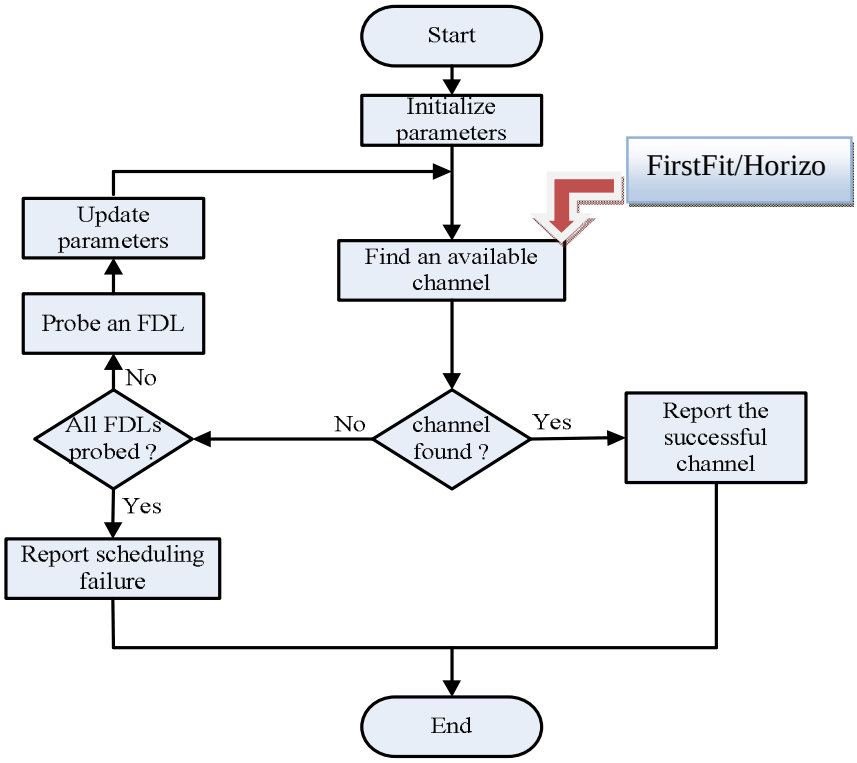
- ***Thuật toán Horizon (LAUC):***

```
unsigned int ndc = totalchannel_ - ncc_;
 int ch = UNAVAILABLE;
   double min_time_gap;
 for( int i = 0; i < ndc; i++ ) {
        double time_gap_ = startTime - horizon_[i];
        if ( time_gap_ >= 0.0 ) {
                if ( ch == UNAVAILABLE ) { // the first time for
updating min_time_gap
```

```
                    min_time_gap = time_gap_;
                    ch = i;
                } else {
                    if ( min_time_gap > time_gap_ ) {
                        min_time_gap = time_gap_;
                        ch = i;
                    }
                } // end of UNAVAILABLE
            } // end of >= 0.0
        } // end of for

        // evaluate the search process to report searching result
        if ( ch != UNAVAILABLE ) {
            result.Fflag() = FOUND;
            result.LambdaID() = ch;
            result.StartTime() = startTime;
        } else {
            result.Fflag() = NOT_FOUND;
        }
        return (result);
    }
```

#### 4.3.3.2 Thuật toán có sử dụng FDL




Hình 4.9 : lưu đồ thuật toán có sử dụng FDL

Đoạn code dùng cho các loại thuật toán có sử dụng bộ đệm FDL giống như không sử dụng bộ đệm, chỉ khác ở chỗ, trước khi cho drop một burst thì biến số starttime sẽ được cộng thêm một lượng là unitdelay, sau đó sẽ là một vòng loop tìm kiếm kênh rỗi lại. Đoạn code cần thêm vào như sau:

```
for( int j = 0; i < N; j++ )
{…
startTime = starTime + unitdelay.
        }
        else {
                result.Fflag() = NOT_FOUND;
         }
           return (result);
        }
```

## 4.5 Kết luận chương

Trong chương này đã trình bày các giải thuật lập lịch trong mạng OBS. Các giải thuật cơ bản là FFUC và LAUC với các trường hợp có hay không sử dụng void filling, trường hợp có hay không sử dụng các đường tạo trễ FDL. Yêu cầu đặt ra là ta phải chọn được giải thuật tốt nhất đáp ứng yêu cầu tối ưu số lượng burst tại đầu vào được sắp xếp trên các kênh dữ liệu để đảm bảo các burst được di chuyển nhanh nhất, đầy đủ nhất đến đầu ra. Trong phần mô phỏng của đồ án sẽ trình bày cụ thể về vấn đề mô phỏng các thuật toán xếp lịch trong mạng OBS, qua đó ta sẽ thấy được tính chất, ưu nhược điểm của từng giải thuật để chon được giải thuật tốt nhất đáp ứng nhu cầu vận chuyển một lượng dữ liệu lớn qua mạng với tốc độ cao. Việc kết hợp các giải thuật cơ bản với sử dụng void filling hay FDL cũng được đề cặp đến trong phần mô phỏng.

**Chương 5**

# MÔ PHỎNG VÀ KẾT QUẢ

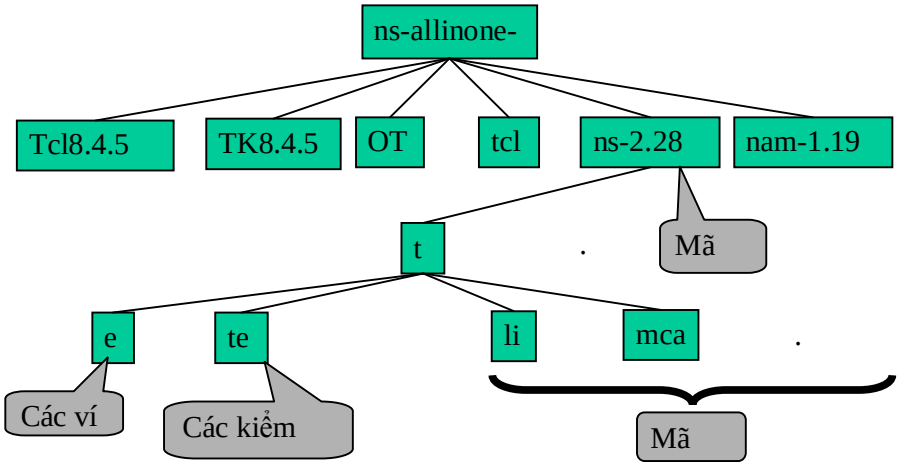
## 5.1 Giới thiệu chương

Trong chương 3 đã trình bày các giải thuật xếp lịch trong mạng OBS. Muốn sắp xếp được càng nhiều burst trên các kênh dữ liệu yêu cầu ta phải chọn được thuật toán tốt nhất để giảm thiểu khả năng mất burst. Đây là một vấn đề rất quan trọng đối với chất lượng của mạng OBS. Đồng thời để giảm khả năng mất burst đến mức thấp nhất có thể ta phải chọn được kích thước burst tối ưu trong quá trình thiết lập burst từ các gói tin riêng rẽ ở đầu vào.Chương này đưa ra kết quả mô phỏng ứng với từng thuật toán được xem xét để chọn được thuật toán nào tốt nhất cho quá trình sắp xếp burst vào các kênh dữ liệu trong mạng OBS. Bên cạnh đó các kết quả mô phỏng cho quá trình thiết lập burst cũng được nêu lên để đánh giá và chọn ra dải kích thước burst trong đó xác suất mất burst là nhỏ nhất đối với mô hình mạng cụ thể trong bài toán mô phỏng. Đồng thời chương này còn giới thiệu sơ lược phần mềm mô phỏng NS2 phục vụ cho mô phỏng các thuật xếp lịch trên.

## 5.2. Giới thiệu phần mềm NS2

Phần mềm NS2(network simulation version 2) là chương trình mô phỏng mã nguồn mở dành cho mục đích nghiên cứu, thực hiện mạng số liệu dựa trên chuyển mạch gói. Không chỉ là công cụ mô phỏng, NS-2 còn là chương trình có nhiều module hỗ trợ và một thư viện rất tiện ích cho việc mô phỏng các sự kiện riêng lẻ. NS2 là chương mô phỏng hướng đối tượng được viết bằng hai ngôn ngữ lập trình C++ và OTcl, chúng hỗ trợ chặt chẽ cho nhau.

- Kiến trúc phần mềm NS2




Hình 5.1. Kiến trúc thư mục cài đặt của NS2 và NAM trong môi trường Linux

Trong số các thư mục con của ns-allinone-2.28 thì ns-2 là nơi chứa các file phục vụ cho mô phỏng (cả viết bằng C++ lẫn OTcl). Trong thư mục này, tất cả OTcl code và những kịch bản ví dụ đều chứa trong thư mục gọi là tcl và hầu hết được viết bằng C. Thư mục tcl có những thư mục con, trong số đó có thư mục lib chứa mã nguồn OTcl cho những thành phần cơ bản nhất và quan trọng nhất (agent, node, link, packet, address, routing,…).

Ns-lib. Tcl: Lớp mô phỏng và đa số các định nghĩa chức năng thành phần của nó ngoại trừ LAN, Web, và Multicast được chứa trong file này.

Ns-default. Tcl: Những giá trị mặc định cho các thong số cấu hình cho các thành phần mạng được chứa ở đây. Bởi vì nhiều thành phần mạng được bổ sung bằng C++, nên những thông số là những biến C++ tạo ra các giá trị cho OTcl qua chức năng liên kết OTcl.

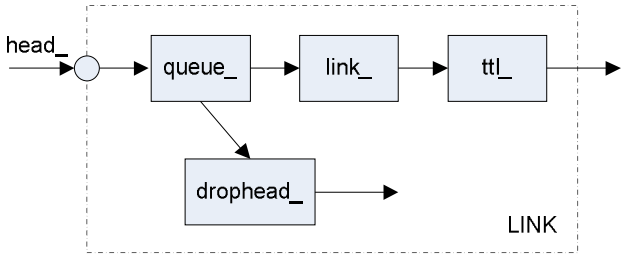
Ns-packet. Tcl: Thành phần khởi tạo những định dạng header của gói được chứa trong file này. Khi tạo ra một gói header thì phải đăng kí header trong file này để tạo ra những xử lý khởi tạo.

Ở mức độ người sử dụng: Việc mô phỏng bắt đầu bằng việc nắm rõ các câu lệnh tạo đối tượng mô phỏng từ đó xây dựng các kịch bản mô phỏng Tcl. Sau khi tạo một kịch bản mô phỏng Tcl, việc chạy chương trình chỉ bằng lệnh trong terminal trong Linux.

Ở mức độ người vừa phát triển vừa sử dụng: Việc phát triển phần mềm bắt đầu từ việc nắm rõ cấu trúc thư mục chính trong NS2 cùng với một số file quan trọng liên quan trọng liên quan đến đối tượng mới cần them vào. Khi người sử dụng tạo ra một chương trình mô phỏng chạy trên nền NS2 cho riêng mình thì cũng có thể tạo đối tượng riêng cho mình, đó cũng là một ưu điểm của phần mềm NS2 mã nguồn mở.

Kiến trúc liên kết của NS2:

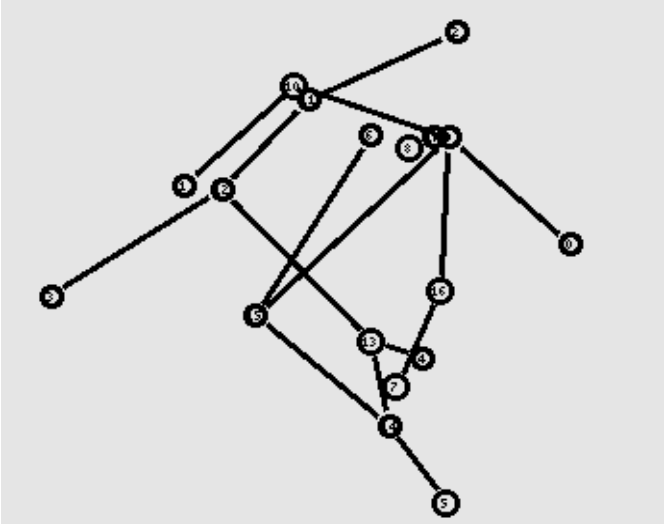
Một bộ phận chính khác trong NS-2 là link. Phần tử kết hợp này gồm ba phần chính: hàng đợi, delay và drophead-object. Hàng đợi quản lý thông lượng các gói phát trên link. Phần tử delay giả lập trễ khi gói truyền trong link. Và drophead-object quản lý các gói bị rớt từ hàng đợi của link. Cấu trúc của link trong NS-2 được mô tả như hình 5.2




Hình 5.2. Kiến trúc liên kết của NS2

## 5.3. Mô phỏng các giải thuật xếp lịch trong mạng OBS

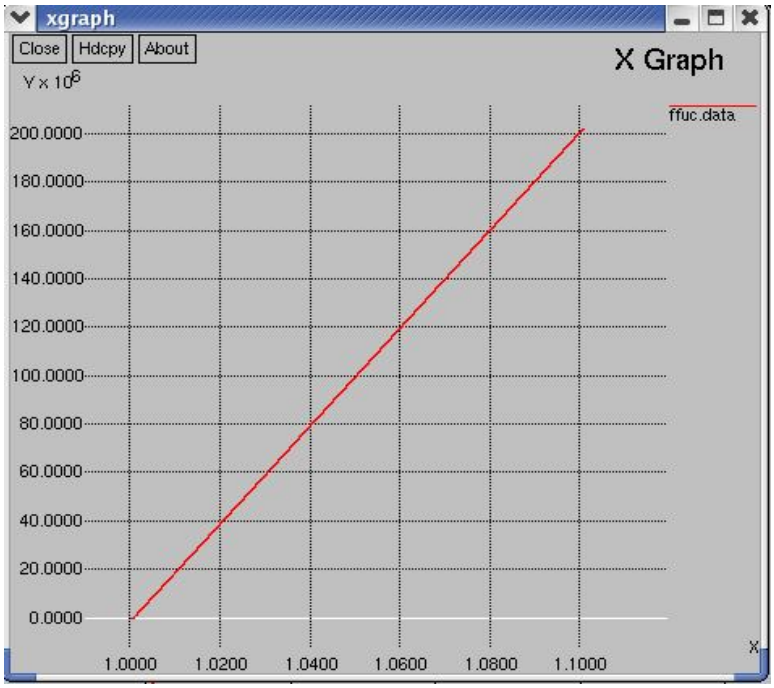
Trong phần mô phỏng sử dụng mô hình mạng gồm 10 node lõi và 10 node biên nối vòng ring như hình. Giao thức được sử dụng là JET với thời gian offset là 0.000001s. Mỗi liên kết có 6 kênh bước sóng gồm 2 kênh điều khiển và 4 kênh dữ liệu. Băng thông mỗi kênh là 20Gb/s. Phương pháp thiết lập burst được sử dụng là thiết lập burst vừa theo độ dài vừa theo thời gian với kích thước tối đa của mỗi burst là 60000 byte, thời gian thiết lập là 0.0003s.



Hình 5.3 Mô hình mạng OBS nối vòng ring

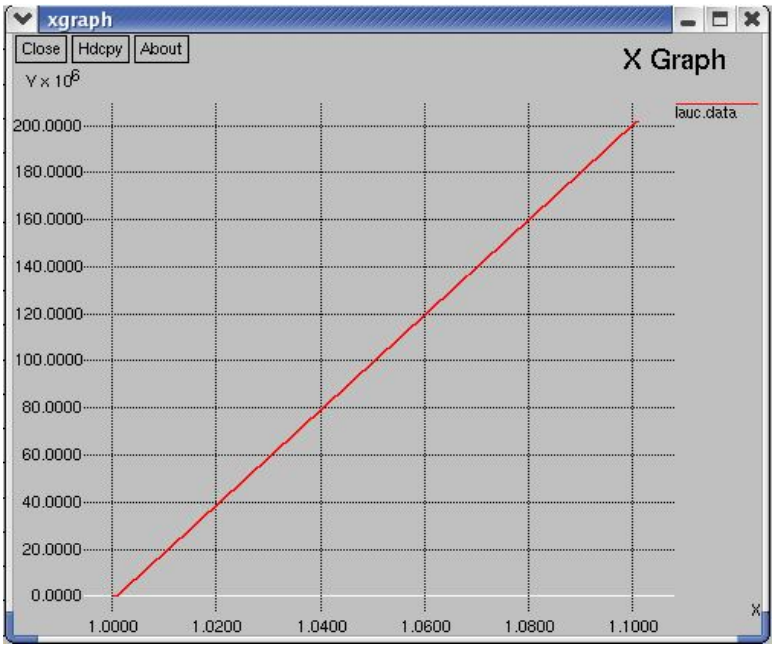
Kết quả cho ra ở mỗi thuật toán là lượng dữ liệu truyền được qua mạng ứng với lưu lượng của mạng thay đổi từ 1.0000 đến 1.1000 Erlang

### 5.3.1 Thuật toán FFUC



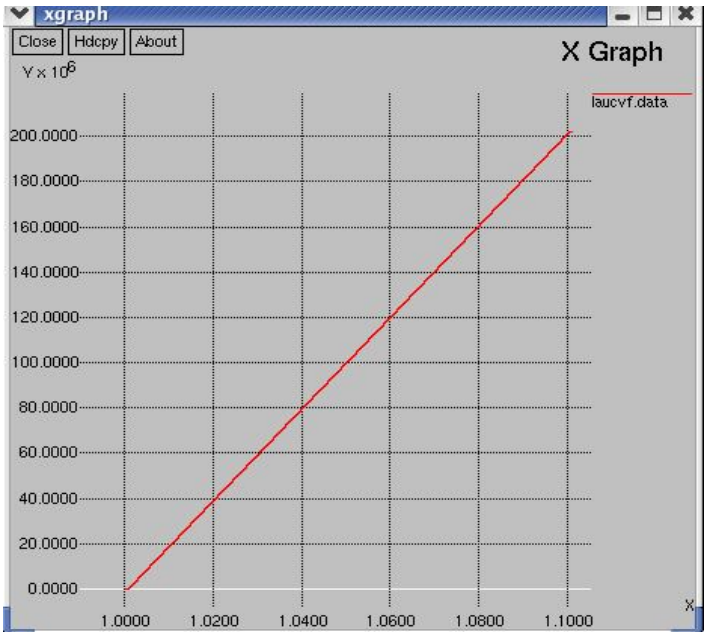

Hình 5.4. Lượng dữ liệu truyền được qua mạng khi sử dụng thuật toán FFUC

### 5.3.2 Thuật toán LAUC




Hình 5.5 Lượng dữ liệu truyền được qua mạng khi sử dụng thuật toán LAUC

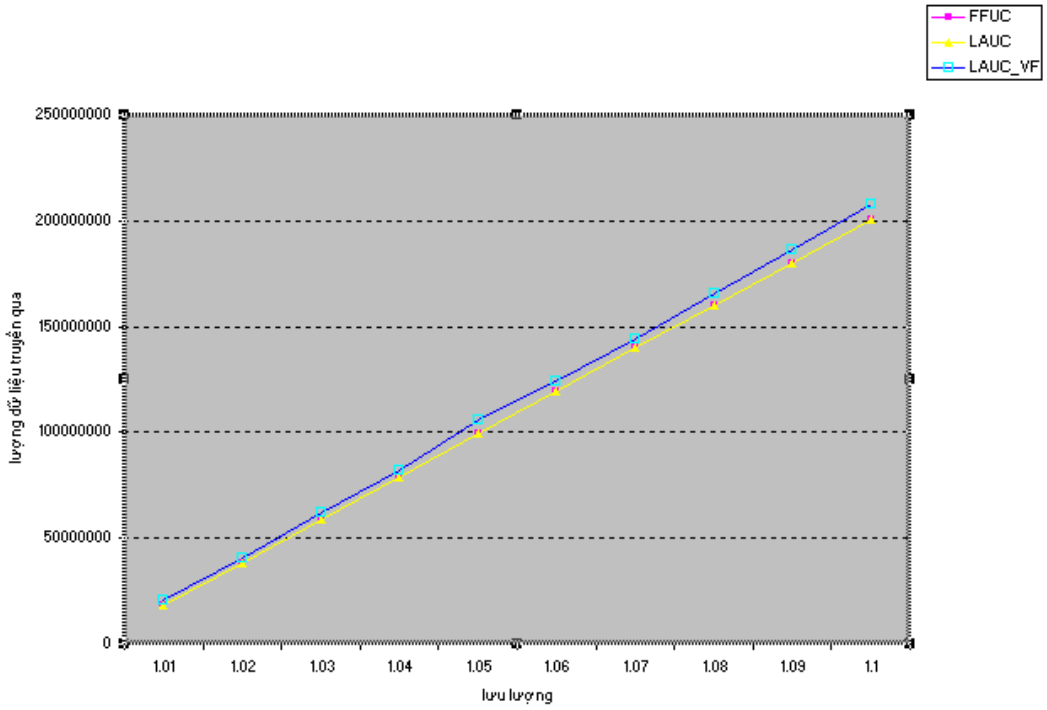
### 5.3.3 Thuật toán LAUC_VF




Hình 5.6 Lượng dữ liệu truyền được qua mạng khi sử dụng thuật toán LAUC-VF

### 5.3.4. So sánh kết quả các thuật toán trên

Để dễ dàng so sánh hiệu quả các thuật toán trên em lấy số liệu kết quả của cả 3 và vẽ trên cùng một đồ thị




Hình 5.7 So sánh lượng dữ liệu truyền qua mạng đối với 3 thuật toán

Dựa vào đồ thị trên ta có thể thấy lượng dữ liệu truyền qua mạng của 2 thuật toán FFUC và LAUC gần như bằng nhau nên 2 đường biểu diễn của chúng trên đồ thị trùng nhau. Trong thực tế thì thuật toán LAUC tuy có sử dụng tài nguyên tốt hơn FFUC do tạo khoảng trống giữa thời gian đến của burst và thời gian sử dụng cuối cùng của kênh dữ nhưng các khoảng trống này khá nhỏ không thể sắp xếp burst khác được nên hiệu quả của thuật toán FFUC và LAUC là như nhau.

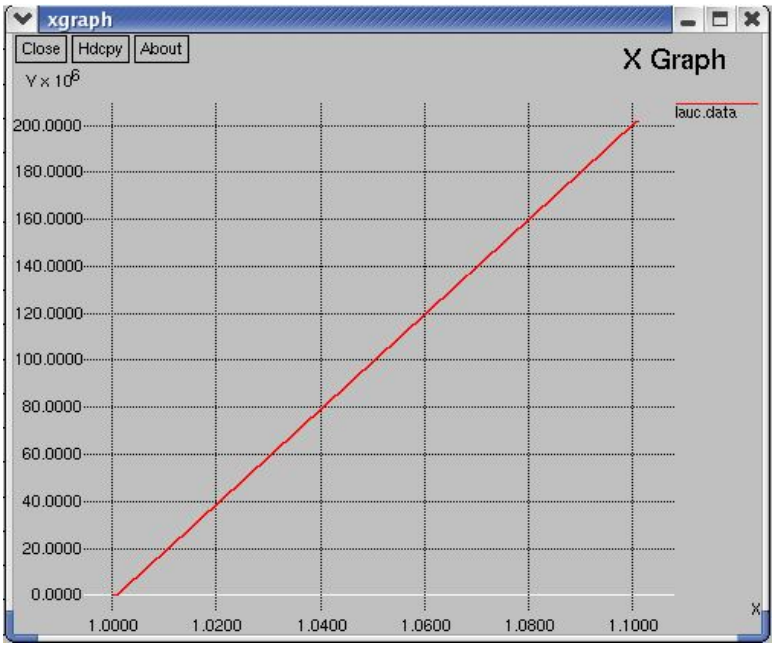
Còn thuật toán LAUC_VF do có xét đến khoảng trống trên các kênh dữ liệu để sắp xếp burst nên hiệu quả cao hơn, lượng dữ liệu truyền được qua mạng cao hơn hẳn 2 thuật toán kia.

Bảng: lượng dữ liệu truyền qua mạng cho các thuật toán

| lưu lượng | lượng dữ liệu truyền | | |
|---|---|---|---|
| | FFUC | LAUC | LAUC_VF |
| 1.01 | 18369000 | 18360000 | 20549000 |
| 1.02 | 38545000 | 38538000 | 40817000 |
| 1.03 | 58775000 | 58770000 | 62068500 |
| 1.04 | 79009000 | 79002000 | 82308000 |
| 1.05 | 99223000 | 99216000 | 105540000 |
| 1.06 | 119504400 | 119503500 | 123845000 |
| 1.07 | 139751000 | 139744500 | 144094000 |
| 1.08 | 159981000 | 159975000 | 165344000 |
| 1.09 | 180189000 | 180180000 | 186603000 |
| 1.1 | 200369000 | 200368500 | 207818500 |

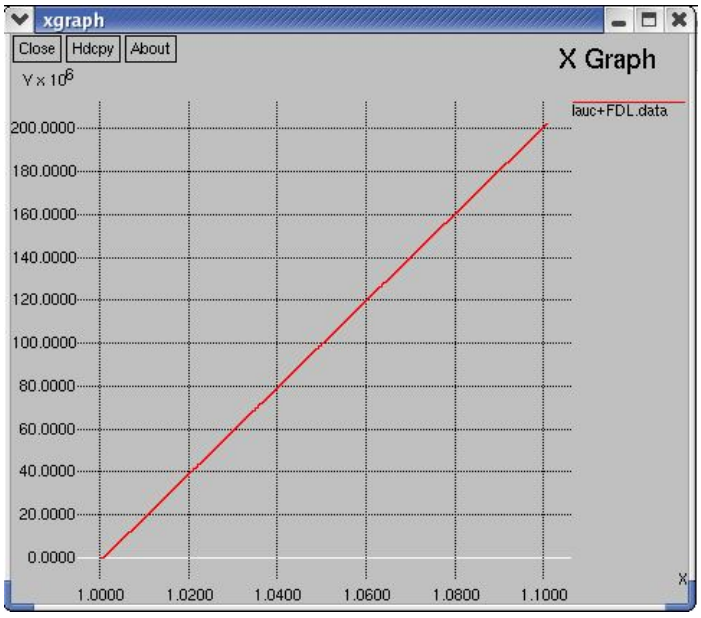
### 5.3.5 So sánh các thuật toán có và không có sử dụng FDL

#### 5.3.5.1 Thuật toán LAUC không sử dụng FDL




Hình 5.8 Lượng dữ liệu truyền qua mạng đối với thuật toán LAUC không sử dụng FDL

### 5.3.5.2 Thuật toán LAUC có sử dụng FDL



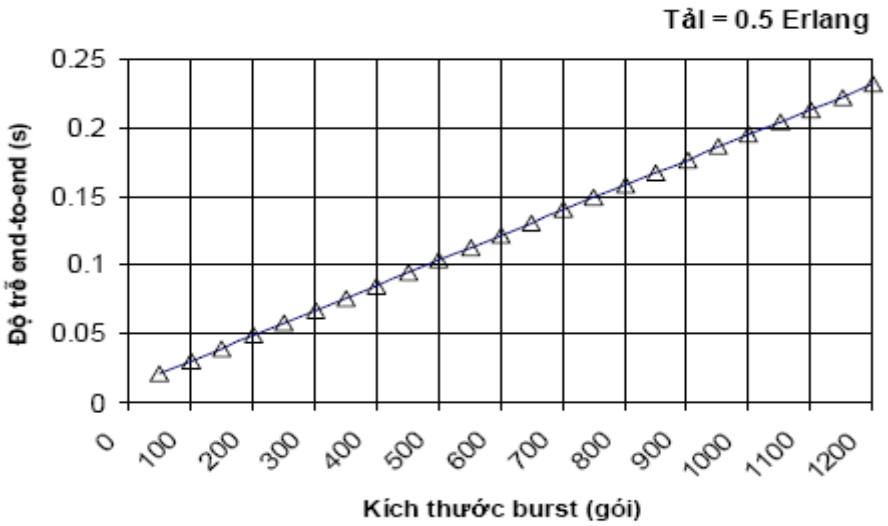

Hình 5.9 Lượng dữ liệu truyền qua mạng đối với thuật toán LAUC có sử dụng FDL

Qua 2 đồ thị biểu diễn kết quả lượng dữ liệu truyền qua mạng khi sử dụng thuật toán LAUC có và không có sử dụng bộ đệm FDL ta thấy việc sử dụng bộ đệm đã làm giảm khả năng mất burst đáng kể, giúp cải thiện khả năng truyền dữ liệu qua mạng rất lớn. Dù gây ra sự tốn kém nhất định nhưng việc sử dụng bộ đệm rất hiệu quả trong mạng OBS vì giúp giảm khả năng mất burst rất cao.

## 5.4. Mô phỏng ảnh hưởng của quá trình thiết lập burst trong mạng OBS (Burst Asembly)

### 5.4.1. Ảnh hưởng của thiết lập burst đến độ trễ trong mạng OBS

Trong mạng OBS các thông số mạng cần chọn một cách hợp lý và các giao thức mạng cần chọn một cách hợp lý sao cho tôt nhất về mặt mất mát và độ trễ.




Hình 5.10 Độ trễ end-to-end trung bình so với kích thước burst

Như hình 5.10 ta thấy kích thước burst tăng lên thì độ trễ end-to end cũng tăng lên theo. Điều này là do kích thước burst tăng lên thì cần thời gian chờ để cho số lượng gói đến đủ để tạo thành một burst. Độ trễ end-to-end ảnh hưởng rất lớn đến các dịch yêu cầu thời gian thực. Với một yêu cầu về độ trễ trung bình thì ta có được giới hạn trên của kích thước burst. Khi có các yêu cầu về dịch vụ và các thông số mạng cho trước ta có thể tìm ra kích thước tối ưu của burst dữ liệu.

### 5.4.2. Bài toán mô phỏng quá trình thiết lập burst

Việc mô phỏng nhằm tìm được một giá trị hay một dải giá trị về kích thước burst cho xác suất mất gói nhỏ nhất trong một mạng OBS với một topo và các thông số mạng liên quan được giới hạn trước.

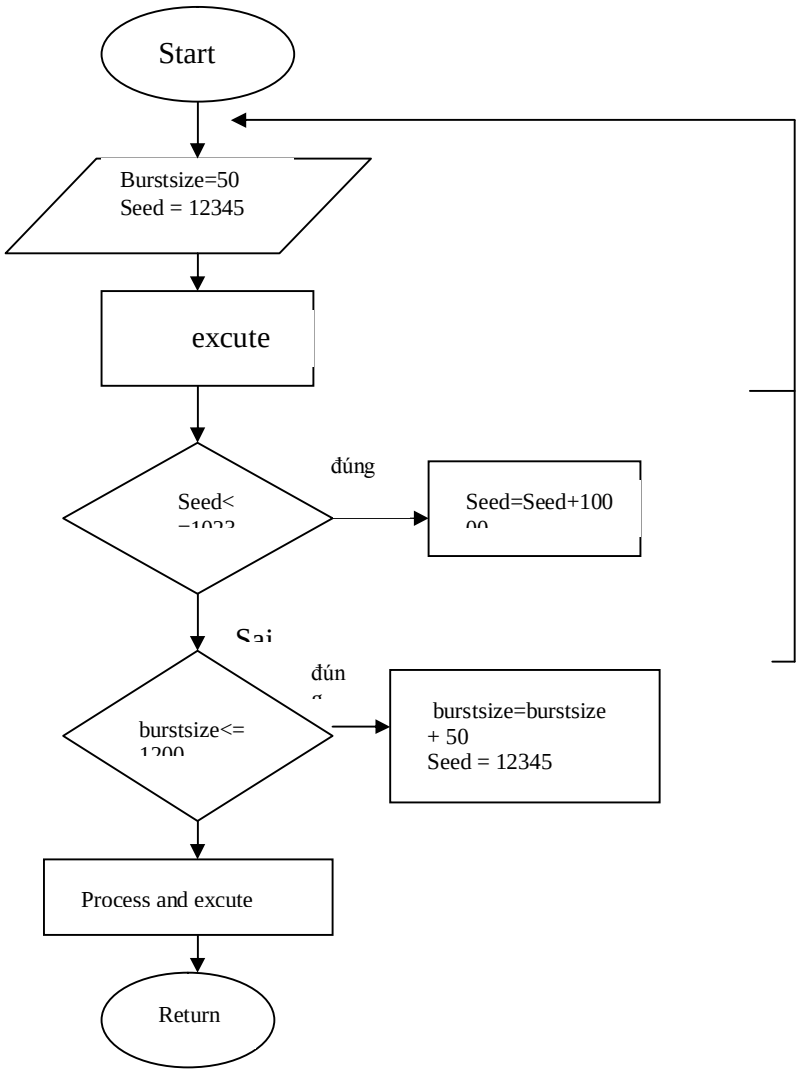
Các thông số giới hạn trong bài toán mô phỏng:

- Mô phỏng mạng NSFNET 14 node, khoảng cách ghi trên hình tính bằng km.
- Các node mạng đều là các node kết hợp
- Liên kết là song hướng, mỗi hướng có hai kênh điều khiển và hai kênh dữ liệu.
- Các gói đến có kích thước cố định là 1250 bytes
- Tốc độ truyền dẫn trên mỗi kênh truyền là 10Gb/s
- Thời gian chuyển mạch là 10 us

- Thời gian xử lý gói điều khiển là 2.5 us
- Lưu lượng được phân bố đồng nhất giữa tất cả các cặp node ngõ vào
- Định tuyến dựa vào đường đi ngắn nhất giữa các cặp node
- Thiết lập burst dựa vào giới hạn tối đa về kích thước burst
- Kích thước gói điều khiển là cố định và bằng 64 bytes
- Giải thuật lập lịch kênh truyền là LAUCVF

Trong mô phỏng so sánh các xác suất mất gói với các mức ngưỡng khác nhau trong khi mạng có và không có phân đoạn burst trong giải quyết xung đột. Mô phỏng bắt đầu với việc xem xét trong mạng chỉ có một lớp dịch vụ (mức ưu tiên) và sau đó là hai lớp dịch vụ.Ta mô phỏng trường hợp: Một mức ngưỡng và có hai ưu tiên.

### 5.4.3. Sơ đồ thuật toán



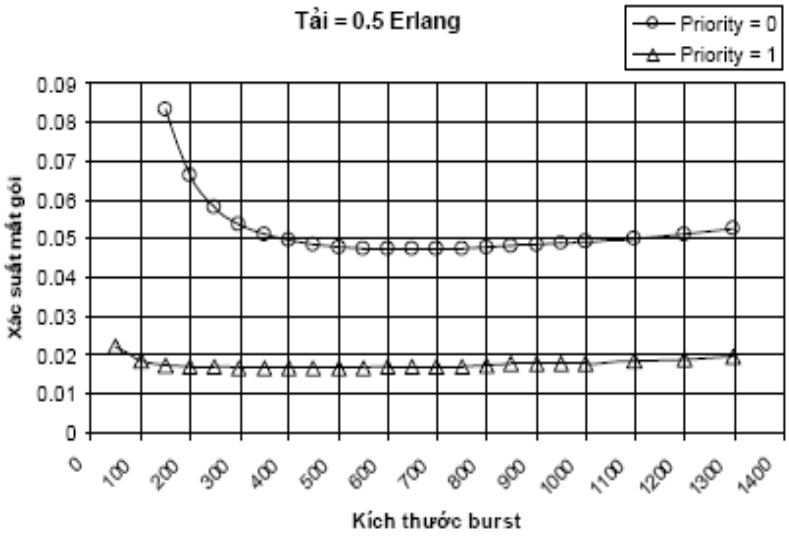

Hình 5.11. Lưu đồ thuật toán mô phỏng

Giải thích lưu đồ thuật toán:

- Burstsize là kích thước burst được tính bằng số lượng gói tin trong một burst dữ liệu.
- Seed được sử dụng để tạo ra một con số ngẫu nhiên cho việc tạo lưu lượng Poisson. Với mỗi số Seed việc phát lưu lượng Poisson sẽ khác nhau.
- Đầu tiên chương trình sẽ gán các thông số ban đầu: burstsize = 50, seed = 12345. Ta sẽ có kết quả về xác suất mất gói ứng với kích thước burst và seed đó. Sau đó seed sẽ được tăng lên 10000 và chạy lần thứ hai, chương trình cứ

tiếp tục như vậy cho đến khi seed đạt giá trị 102345. Sau đó ta lấy giá trị trung bình của xác suất mất burst tức là ta đã có một điểm trên trên đồ thị ứng với kích thước burst là 50. Ta lần lượt tăng kích thước burst lên 50 cho đến khi đạt 1200 gói tin/burst. Như vậy ta được đồ thị mô tả về xác suất mất burst ứng với kích thước burst từ 50 đến 1200 gói tin/ burst.

### 5.4.4. Trường hợp một mức ngưỡng có hai mức ưu tiên




Hình 5.12 Xác suất mất gói của từng mức dịch vụ đối với các kích thước burst khác nhau.

Ta có hai mức ưu tiên là mức ưu tiên số 1 cao hơn và mức ưu tiên số 0 là mức ưu tiên nhỏ hơn. Theo hình ta có đối với lớp dịch vụ cao hơn thì kích thước burst tối ưu là từ 200 đến 700 còn đối với mức ưu tiên thấp hơn thì kich thước burst tối ưu là 550 đến 700. Lớp dịch vụ có mức ưu tiên cao hơn sẽ cho xác suất mất gói thấp hơn.

## 5.5 Kết luận chương

Như vậy là trong chương này em đã tiến hành mô phỏng được các thuật toán lập lịch và nêu kết quả tối ưu quá trình thiết lập burst.

Đối với việc mô phỏng các thuật toán lập lịch em đã lấy kết quả vẽ trên cùng một đồ thị lượng dữ liệu truyền qua mạng của các thuật toán sắp xếp kênh dữ liệu trong mạng OBS gồm FFUC, LAUC, LAUC_VF để thấy được ưu điểm của

thuật toán LAUC_VF so với 2 thuật toán kia. Phần kết quả đối với thuật toán có sử dụng và không sử dụng FDL cũng được trình bày. Qua đó ta thấy được việc sử dụng FDL đã làm giảm lượng burst mất đi đáng kể, từ đó làm tăng lượng dữ liệu được truyền đi. Vậy thuật toán tối ưu là thuật toán LAUCVF kết hợp với các đường trễ quang FDL sẽ cho kết quả tốt nhất.

Từ kết quả mô phỏng của bài toán tối ưu trong quá trình thiết lập burst cho thấy tồn tại một kích thước burst cho xác suất mất burst là nhỏ nhất. Còn kích thước burst là bao nhiêu thì tùy thuộc vào topo mạng và các thông kèm theo. Trong đồ án em chỉ xét trường hợp các gói đến có kích thước cố định nên kích thước burst cũng được tính bằng số lượng gói của nó.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI

Trong hoàn cảnh các dịch vụ mạng đang phát triển mạnh mẽ với nhiều loại hình phong phú, đa dạng không chỉ dừng lại ở các dịch vụ tốc độ cố định như thoại hay truyền hình mà còn phục vụ nhiều cho truyền dữ liệu với tốc độ thay đổi. Vì vậy đặt ra yêu cầu phải có một công nghệ đáp ứng được tính chất đột biến của lưu lượng trong mạng. Và chuyển mạch chùm quang OBS là sự lựa chọn ưu việt nhất khi các công nghệ hiện đại đáp ứng cho chuyển mạch gói quang OPS như bộ đệm quang hay logic quang vẫn chưa thực hiện được. Dù không có nhiều ưu điểm nổi bật như OPS nhưng OBS cũng có thể truyền được một lượng dữ liệu với tốc độ cao và là một công nghệ hứa hẹn của mạng đường trục thế hệ sau.

Mục đích của đồ án là giới thiệu tổng quát về chuyển mạch chùm quang, trong đó tìm hiểu sâu hơn và tiến hành mô phỏng các giải thuật xếp lịch trong mạng OBS cũng như tìm ra kích thước burst tối ưu để giảm thiểu tỉ lệ mất burst. Qua kết quả mô phỏng 2 vấn đề trên có thể kết luận rằng thuật toán tối ưu cho việc sắp xếp kênh dữ liệu là LAUC_VF kết hợp với việc sử dụng bộ đệm FDL vì cho lượng dữ liệu truyền qua mạng cao nhất so với các thuật toán còn lại. Đồng thời ta thấy được đối với topo mạng trong phần mô phỏng trên thì kích thước burst nằm trong khoảng 350-750 gói cho xác suất mất burst nhỏ nhất đối với trường hợp một mức ngưỡng và không có mức ưu tiên. Còn trường hợp một mức ngưỡng và có 2 mức ưu tiên thì số gói 550-750 đối với mức ưu tiên số 0 và số gói 200-700 đối với mức ưu tiên số 1 trong một burst là tối ưu. Như vậy đối với mỗi mô hình mạng cụ thể luôn tồn tại một dải kích thước burst xác định cho xác suất mất burst nhỏ nhất.

Hướng phát triển của đề tài là xây dựng một mô hình mạng phức tạp hơn với việc tăng số lượng node, số lượng các bộ scheduler. Từ đó tiến hành mô phỏng với nhiều thuật toán khác, kết hợp 2 phương thức sắp xếp burst dựa trên mức ngưỡng (chiều dài burst) và bộ định thời (thời gian sắp xếp) để nâng cao chất lượng cho mạng OBS.

# TÀI LIỆU THAM KHẢO

[1] Banks, J., Carson, J., Nelson, B., Nicol, D., (2001) Discrete-Event System Simulation, Third Edition, New Jersey: Prentice-Hall

[2] Battestilli, T., and Perros, H., (2003), An Introduction to optical burst switching IEEE Communications Magazine, vol .41, no .8, pp .S10-S15

[3] Biao Chen, Vinod M. Vokkarane, Jason P.Ju, "Absolute QoS Differentiation in Optical Burst Switched Network", IEEE, 2004

[4] Chen, Y.,Qiao, c and Yu ,X., (2004), Optical burst switching: a new area in optical networking research, IEEE Network, vol .18, pp.16-23

[5] Chunming Qiao, "Optical Burst Switching OBS – A new paradigm for a Optical Internet", IEEE

[6] Fall K and Varadhan K., (2005), The ns Manual, UC Berkeley, LBNL, USC/ISI, and Xerox PARC, Jan., <URL: http://www.isi.edu/nsnam/ns/ns-documentation.html>

[7] Gauger, C.M. (2003), Trends in Optical Burrst Switching, Proceedings of SPIE ITCOM 2003, Orlando

[8] Hai Le Vu and Moshe, "Blocking Probability for Priority Classes in Optical Burst Switching Network", IEEE

[9] Jinhui Xu, Qiao, Jikai Li and Guang Xu, "Efficient Channel Scheduling Algorithms in Optical Burst Switched Network", IEEE, 2003

[10] Martin Nord, "Optical switching technology for optical burst and packet switch", 2002

[11] M. Klinkowski ,"Performance Analysis of Isolated Adaptice Routing in OBS network", Advanced Broadband communication Center, 2005.

# PHỤ LỤC

**Mã nguồn các giải thuật xếp lịch**

```
#Các thông số ban đầu
#10CNs in ring; 10 attached ENs
#20 Gbit/s channels
StatCollector set debug_ 0
Classifier/BaseClassifier/EdgeClassifier set type_ 0
Classifier/BaseClassifier/CoreClassifier set type_ 1
#Thời gian xử lý gói tin điều khiển tại một node là 1us
source ../lib/ns-obs-lib.tcl
source ../lib/ns-obs-defaults.tcl
source ../lib/ns-optic-link.tcl
set ns [new Simulator]
set nf [open basic01.nam w]
set sc [new StatCollector]
set tf [open trace01.tr w]
set ndf [open ndtrace01.tr w]
set old_data 0
# dump all the traces out to the nam file
$ns namtrace-all $nf
$ns trace-all $tf
$ns nodetrace-all $ndf
#=============================================================
=========#
# các thông số bất biến trong mạng
BurstManager offsettime   0.00001
#kích thước burst tối đa
BurstManager maxburstsize 60000
```

```
#thời gian thiết lập burst
BurstManager bursttimeout 0.0003
Classifier/BaseClassifier/CoreClassifier set bhpProcTime 0.000001
Classifier/BaseClassifier/EdgeClassifier set bhpProcTime 0.0000015
#giả sử có 1 FDL trên mỗi kênh bước sóng đầu ra
Classifier/BaseClassifier set nfdl 10
Classifier/BaseClassifier set fdldelay 0.0001
Classifier/BaseClassifier set option 0
Classifier/BaseClassifier set maxfdls 8
Classifier/BaseClassifier set ebufoption 1
#this is a fixed delay line present at the ingress of every node
OBSFiberDelayLink set FDLdelay 0.0001
# total number of edge nodes
set edge_count 10
# total number of core routers
set core_count 10
# total bandwidth/channel (1mb = 1000000)
set bwpc 20000000000
#set bwpc
# delay in milliseconds
set delay 0.02ms
# tổng số kênh bước sóng trên 1 liên kết
set maxch 4
# số kênh điều khiển
set ncc 1
# số kênh dữ liệu
set ndc 3
```

```
#=========================================================
=========#
# các quá trình hỗ trợ
# finish procedure
proc finish {} {
   global ns nf sc tf ndf old_data
   $ns flush-trace
   $ns flush-nodetrace
   close $nf
   close $tf
   close $ndf
      $sc display-sim-list
   #Execute NAM on the trace file
   #exec nam basic01.nam &
   exec awk {
    {
   if ( $1=="r") {
   old_data = old_data + $5 }
      print $2, old_data*1.0
   }
} ndtrace01.tr > ffuc.data
   exec xgraph ffuc.data &
   puts "Simulation complete";
   exit 0
}
 #print $2, old_data*8.0/$2/10000000000
#tạo ra topo node biên-node lõi-node biên
Simulator instproc  create_topology { } {
   $self instvar Node_
```

```
global E C
global edge_count core_count
global bwpc maxch ncc ndc delay
set i 0
# thiết lập node biên
while { $i < $edge_count } {
      set E($i) [$self create-edge-node $edge_count]
   set nid [$E($i) id]
   set string1 "E($i) node id:     $nid"
   puts $string1
      incr i
}
   set i 0
# thiết lập node lõi
while { $i < $core_count } {
      set C($i) [$self create-core-node $core_count]
   set nid [$C($i) id]
   set string1 "C($i) node id:     $nid"
   puts $string1
      incr i
}
   $self createDuplexFiberLink $E(0) $C(0) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(1) $C(1) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(2) $C(2) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(3) $C(3) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(4) $C(4) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(5) $C(5) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(6) $C(6) $bwpc $delay $ncc $ndc $maxch
$self createDuplexFiberLink $E(7) $C(7) $bwpc $delay $ncc $ndc $maxch
```

```
    $self createDuplexFiberLink $E(8) $C(8) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $E(9) $C(9) $bwpc $delay $ncc $ndc $maxch


    $self createDuplexFiberLink $C(0) $C(1) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(1) $C(2) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(2) $C(3) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(3) $C(4) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(4) $C(5) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(5) $C(6) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(6) $C(0) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(7) $C(0) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(8) $C(0) $bwpc $delay $ncc $ndc $maxch
    $self createDuplexFiberLink $C(9) $C(0) $bwpc $delay $ncc $ndc $maxch
      $self build-routing-table
    }
#create a self-similar traffic-stream over a UDP agent
Simulator instproc  create_selfsim_connection { selfsim udp null src dest start0 stop0 } {
    upvar 1 $udp udpr
    upvar 1 $selfsim selfsimr
    upvar 1 $null nullr
    upvar 1 $src srcr
    upvar 1 $dest destr
    set udpr [ new Agent/UDP]
    $self attach-agent $srcr $udpr
    set selfsimr [ new Application/Traffic/SelfSimilar ]
    $selfsimr set starttime $start0
    $selfsimr set stoptime $stop0
```

```
    $selfsimr attach-agent $udpr
    set nullr [ new Agent/Null ]
    $self attach-agent $destr $nullr
    $self connect $udpr $nullr

    $self at $start0 "$selfsimr start"
    $self at $stop0 "$selfsimr stop"
   puts "traffic stream between $src = $srcr and $dest = $destr created"
}
$ns create_topology
Agent/UDP set packetSize_ 1500
Application/Traffic/SelfSimilar set batchsize 4500
Application/Traffic/SelfSimilar set sb 0
Application/Traffic/SelfSimilar set Hb -0.5
Application/Traffic/SelfSimilar set rate 5000.0
Application/Traffic/SelfSimilar set std_dev_inter_batch_time 1.0e-5
Application/Traffic/SelfSimilar set Ht 0.5
#add traffic stream between every pair of edge nodes in both directions
set i 0
 while {$i < $edge_count} {
  set j 0
    while {$j < $edge_count} {
        if {$i != $j} {
      $ns  create_selfsim_connection  selfsim($i:$j) udp($i:$j) null($i:$j) E($i) E($j)
1.0 1.1
            }
     incr j
  }
   incr i
```

```
}
$ns at 5.1 "finish"
$ns run
```

**Mã nguồn thiết lập burst**

```
set number [lindex $argv 0]
set opt(seed) [lindex $argv 1]

# source for simulation
source /home/ext/ns-allinone-2.28/ns-2.28/obs4ns-3.4/tcl/lib/ns-obs-lib.tcl
source /home/ext/ns-allinone-2.28/ns-2.28/obs4ns-3.4/tcl/lib/ns-obs-node.tcl
source /home/ext/ns-allinone-2.28/ns-2.28/obs4ns-3.4/tcl/lib/ns-obs-link.tcl
source /home/ext/ns-allinone-2.28/ns-2.28/obs4ns-3.4/tcl/lib/ns-obs-stats.tcl
source /home/ext/ns-allinone-2.28/ns-2.28/obs4ns-3.4/tcl/lib/ns-obs-defaults.tcl

set ns [new Simulator]

global defaultRNG
$defaultRNG seed $opt(seed)

Connector/ObsLink dc_bandwidth 10Gb
Connector/ObsLink cc_bandwidth 10Gb
Agent/Burstifier set max_db_size_          [expr 1250*$number]
Agent/Burstifier set bhp_size_             64
Agent/Burstifier set timeout_              2000ms
Agent/OXC switch_time       10us
Agent/SCU max_bhp_proc_time     2.5us
Agent/Burstifier set max_packets_          2000
[Agent/SCU set bhp_proc_time_] set max_ [Agent/SCU max_bhp_proc_time]
Agent/Burstifier set max_segmentations_     10
```

```
Agent/Burstifier set min_segmentable_size_  1250
Agent/Burstifier set segmentation_          true
Agent/SCU max_segmentations 10
Agent/SCU min_segmentable_size 1250
Agent/SCU set segmentation_  10
Agent/SCU set deflection_    0

set edge_count 14
set core_count 14
set ndc 1
set ncc 1
set n_app 364
set n_links 21
set stype LAUCVF
set load 0.5

# # set up the hybrid nodes
set i 1
   while { $i <= $core_count } {
        set c($i) [$ns ObsHybridNode $ncc $ndc ChannelScheduler/$stype 2]
        incr i
   }

#creat NSFNET in real distances
$ns duplex-obs-link $c(1) $c(2) $ncc $ndc 1100 ChannelScheduler/$stype
$ns duplex-obs-link $c(1) $c(3) $ncc $ndc 1600 ChannelScheduler/$stype
$ns duplex-obs-link $c(1) $c(8) $ncc $ndc 2800 ChannelScheduler/$stype
$ns duplex-obs-link $c(2) $c(3) $ncc $ndc 600 ChannelScheduler/$stype
$ns duplex-obs-link $c(2) $c(4) $ncc $ndc 1000 ChannelScheduler/$stype
```

```
$ns duplex-obs-link $c(3) $c(6) $ncc $ndc 2000 ChannelScheduler/$stype
$ns duplex-obs-link $c(4) $c(5) $ncc $ndc 600 ChannelScheduler/$stype
$ns duplex-obs-link $c(4) $c(11) $ncc $ndc 2400 ChannelScheduler/$stype
$ns duplex-obs-link $c(5) $c(6) $ncc $ndc 1100 ChannelScheduler/$stype
$ns duplex-obs-link $c(5) $c(7) $ncc $ndc 800 ChannelScheduler/$stype
$ns duplex-obs-link $c(6) $c(10) $ncc $ndc 1200 ChannelScheduler/$stype
$ns duplex-obs-link $c(6) $c(13) $ncc $ndc 2000 ChannelScheduler/$stype
$ns duplex-obs-link $c(7) $c(8) $ncc $ndc 700 ChannelScheduler/$stype
$ns duplex-obs-link $c(8) $c(9) $ncc $ndc 700 ChannelScheduler/$stype
$ns duplex-obs-link $c(9) $c(10) $ncc $ndc 900 ChannelScheduler/$stype
$ns duplex-obs-link $c(9) $c(12) $ncc $ndc 500 ChannelScheduler/$stype
$ns duplex-obs-link $c(9) $c(14) $ncc $ndc 500 ChannelScheduler/$stype
$ns duplex-obs-link $c(11) $c(12) $ncc $ndc 800 ChannelScheduler/$stype
$ns duplex-obs-link $c(11) $c(14) $ncc $ndc 800 ChannelScheduler/$stype
$ns duplex-obs-link $c(12) $c(13) $ncc $ndc 300 ChannelScheduler/$stype
$ns duplex-obs-link $c(13) $c(14) $ncc $ndc 300 ChannelScheduler/$stype
$ns compile-obs


set rate [expr $load*$ndc*[Connector/ObsLink dc_bandwidth]/$n_app]
#k refer to class of service
set k 0
while {$k < 2} {
set i 1
 while {$i <= $edge_count} {
   set j 1
   while {$j <= $edge_count} {
         if {$i != $j} {
#the rate of one class is a half
```

```
set Poi($i$j$k) [new Application/Traffic/Poisson]
$Poi($i$j$k) set rate_ $rate
$Poi($i$j$k) set packetSize_ 1250
#creat udp to attach traffic
set udp($i$j$k) [$c($i) set burstifier_([$c($j) id]:$k)]
$Poi($i$j$k) attach-agent $udp($i$j$k)
$udp($i$j$k) set-traffic-generator $Poi($i$j$k)
$ns at 0.0 "$udp($i$j$k) start"

    }
     incr j
  }
    incr i
}
incr k
}



proc stop {} {
global ns udp edge_count
for { set k 0 } { $k <2} { set k [expr $k + 1]} {
for { set i 1 } { $i <= $edge_count} { set i [expr $i + 1]} {
          for { set j 1 } { $j <= $edge_count} { set j [expr $j + 1]} {
             if { $i != $j } {
          $ns at-now "$udp($i$j$k) stop"
             }
          }
}
```

```
}
set now [$ns now]
$ns at [expr $now + 0.2] "finish"

}
# finish procedure
proc finish {} {
global ns sc0 sc1 defaultRNG number

set ip_snd0 [expr [$sc0 get-counter-value DATA_SND]]
set ip_rcv0 [expr [$sc0 get-counter-value DATA_RCV]]
set ip_drop0 [expr $ip_snd0 - $ip_rcv0]
set ip_p0 [expr 1.0*$ip_drop0/$ip_snd0]

set file0 [open "results-0.txt" "a"]
puts $file0 "$ip_p0"


set ip_snd1 [expr [$sc1 get-counter-value DATA_SND]]
set ip_rcv1 [expr [$sc1 get-counter-value DATA_RCV]]
set ip_drop1 [expr $ip_snd1 - $ip_rcv1]
set ip_p1 [expr 1.0*$ip_drop1/$ip_snd1]
set file1 [open "results-1.txt" "a"]
puts $file1 "$ip_p1"
exit 0
}
set sc0 [$ns get-global-stats-collector 0]
$sc0 set-counter-convergence DATA_SND 1250000
Stats stop-command "stop"
```

```
set sc1 [$ns get-global-stats-collector 1]
$sc1 set-counter-convergence DATA_SND 1250000
Stats stop-command "stop"
# enable stats collector
$ns at [RouteLogic/ObsRoute transit_time] "$ns enable-stats"
$ns run
```