



---

## **Optimized Upstream MAC-Scheduling for Broadband Cable Networks**

*Yingfei Dong, Zhi-Li Zhang, and David H.-C. Du*

Computer Networking and Multimedia Research Lab

Dept. of Computer Science and Engineering,

University of Minnesota

**dong, zhzhang, du@cs.umn.edu**



---

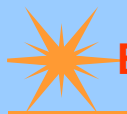
## **Outline**

- Background of MAC scheduling in DOCSIS
- Analysis of the relation between map frequency and channel utilization
- Formulation of optimal scheduling problem
- Optimal Data-Slot placement algorithm

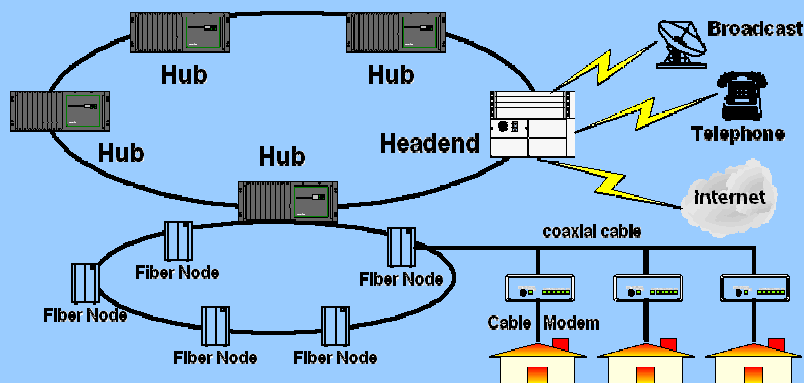


## Motivations and Background

- Wide Deployment of Broadband Service
  - Cable Modem (CM), DSL, Fixed-Wireless, and Satellite
  - High BW to home enables many new applications
    - VOD (1~ 3 Mbps), Interactive Gaming(<200Kbps), Video-phone(<100Kbps, 150 ms), etc.
- Cable Broadband Service
  - Broad Infrastructure: Cable networks cover 90% of homes in US
  - High BW: Downstream 27Mbps / Upstream 2.5 Mbps per channel.  
1.5 Mbps / 128Kbps per CM
  - **Shared Cable Channels** v.s. DSL's dedicated links
  - Dominant MAC protocol
    - Data Over Cable Service Interface Specification (DOCSIS)
    - IEEE 802.14

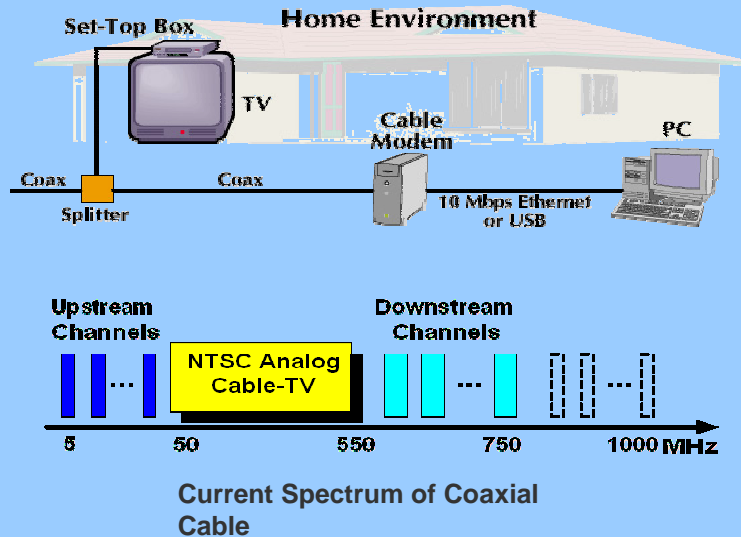


## Broadband Cable Networks (BCNs)





## Cable Technology



## Limitation and Challenge

### ➤ Current Issues

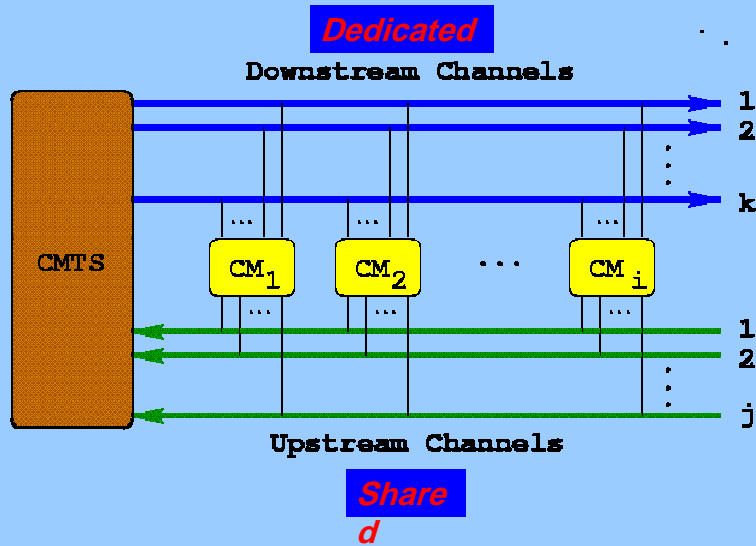
- **Asymmetry in downstream/upstream capacity**
  - Upstream BW is limited: A CM can access only a few upstream channels
  - Upstream packet delay / loss is irregular due to competition among CMS
- **DOCSIS 1.1 defines only reserved-based service classes**
- **No complete systematical study on DOCSIS MAC scheduling has been published although the upstream MAC scheduling is open to vendors**
  - Few studies on DOCSIS, mostly focusing on contention resolution; Scheduling delay is not discussed in depth
  - Other studies on IEEE 802.14

### ➤ Challenges:

- How to efficiently utilize limited upstream BW
- How to effectively support two-way applications
  - BW-sensitive or Delay-sensitive Applications*

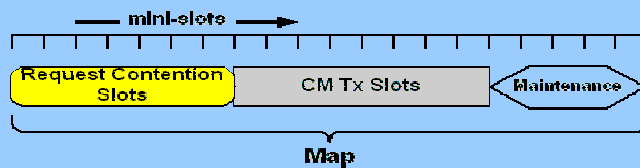


## System Architecture



## Upstream MAC Protocol

- An upstream channel is divided as a stream of **mini-slots**



- Allocation Map:

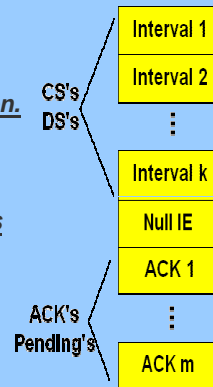
- Periodically broadcast on a downstream channel
  - a fixed-length header + Information Elements (IEs)
- Allocation IEs:
  - Request IE, Data Grant IE, Request/Data IE.
- Other IEs: Data ACK IE, Maintenance IE, etc.



## Upstream MAC Protocol (Cont'ed)

### ➤ Each CM scans the map for available slots

- **Contention Slots (CS's) :**  
Request IEs, Request/Data IEs  
– These slots are subject to competition/contention.
- **Data Slots (DS's):**  
Data Grant IEs for individual CM  
– These slots are dedicated to individual CMs
- **Piggybacking**  
– A request carried the next outgoing data frame
- Only one request outstanding per Service ID



### ➤ Scheduling requests at CMTS is open to vendors



## Collision Resolution Algorithm

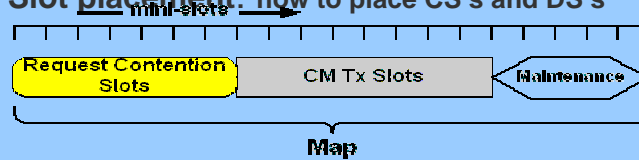
### ➤ Truncated binary exponential back-off (similar to Ethernet)

- initial and maximum back-off window set by CMTS in a MAP.
- When a CM need to send a request frame, it
  - sets its back-off window equal to the initial back-off window.
  - select a random number  $W$  within this window.  
( $W$  is the number of CS's that the CM must defer before transmitting.)
- CM waits for a Data Grant/Data Grant Pending (for contention request) or for an ACK (for data contention) after a transmission.
- Either is received, the contention resolution is complete. Otherwise, contention occurs.
  - multiply its back-off window by two (until max back-off window)
  - restart the process described above.



## Issues in Upstream Scheduling

- Map frequency/size: how often to send a map
  - Fast  $\Rightarrow$  higher management cost, low efficiency
  - Slow  $\Rightarrow$  longer frame delay
- Slot Ratio of CS's to DS's in a map (CS = 16 B, DS  $\leq$  1500 B)
  - Many CS's  $\Rightarrow$  less contention, potential waste of BW
  - Few CS's  $\Rightarrow$  longer request access delay, waste of DS's
  - Ideal Case : number of CS's  $\sim$  number of requests
    - Estimate the number of requests during a map interval.
- Slot placement: how to place CS's and DS's



## Corresponding studies

- Analysis of the relation between the map frequency and channel utilization
  - Simple bounds
- Formulation the optimal scheduling problem
  - Approximate Algorithms
- Optimal Data-Slot (DS) Placement Scheme
  - Splitting-DS improves the chances of piggybacking



## Analysis of Map frequency

➤ **Upstream Frame Delay:** contention delay + scheduling Delay

Contention Delay: request contention and backoff delay

Scheduling Delay: scheduling algorithm at CMTS

➤ **Goal**

control mean scheduling delay and channel utilization

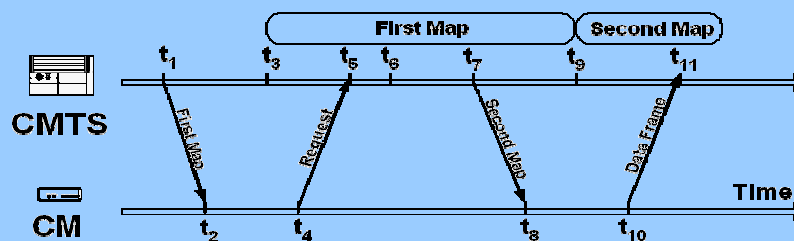
➤ **Assumptions**

Focus on the scheduling delay. Assume each request passes the contention process

All frames have the same size L



## Delivery of an Upstream Frame



A CM sends a request in a CS of 1<sup>st</sup> Map

- No contention
- If Data Granted in 2<sup>nd</sup> Map
- CM sends a data frame in granted DS's of 2<sup>nd</sup> Map
- Worst-Case: CMTS immediately schedules a map after recv a request



## Worst-Case Scheduling Delay and Utilization

$$D = c_1 + c_2 + c_3 + d_{req\_up} + d_{mapdown} + d_{frameup} + d_{compete} \approx C + \frac{L}{B_{up}}$$

where  $c_1$  is the mean random offset chosen by the algorithm,

$c_2$  is the CM map-scanning delay (0.2 ms),

$c_3$  is the map-generating delay at CMTS,

$d_{req\_up} \approx d_{mapdown} \approx d_{frameup} \approx$  One-way propagation delay,

$d_{compete} \approx 0$

$C$  is the sum of all constants,  $B_{up}$  is the channel BW.

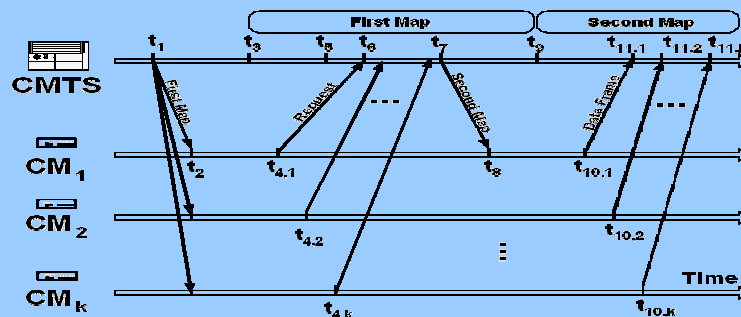
$$\rho_1 \approx \frac{1}{\frac{B_{up}}{L} \cdot C + 1} = 0.24$$

when  $C = 0.3$  ms,  $L = 300$  B,  
and  $B_{up} = 2.5$  Mbps.



## Grouping $k$ requests in a map

- CMTS waits  $d_{map}$  seconds
- Assume requests arrive at CMTS as Poisson with  $\lambda$  req/sec.
- Group  $k = d_{map} \cdot \lambda$  requests in a map





## Better Channel Utilization

$$D_k = \sum_{i=1}^k d_i \approx C + d_{map} + \frac{k \cdot L}{B_{up}} \quad D_{mean} = \frac{D_k}{k} \approx \frac{C + d_{map}}{k} + \frac{L}{B_{up}}$$

where  $d_i$  is the delay of  $i^{th}$  frame,  $d_{map}$  is the map delay,  $D_k$  is the sum of delays of all  $k$  frames, and  $D_{mean}$  is the mean delay of frames.

$$\rho \approx \frac{1}{\frac{B_{up}}{L} \cdot \frac{C + d_{map}}{\lambda \cdot d_{map}} + 1} = 0.42$$

when  $\lambda = 1000$ ,  $d_{map} = 0.01$ , and Other parameters are the same.



## Relation among delay, $\eta$ and $d_{map}$

- Given a mean delay requirement  $D_r$  and an expected channel utilization  $\eta$  we need to ensure

$$d_{map} \leq \frac{C}{(D_r \cdot \frac{B_{up}}{L} - 1) \cdot \eta - 1}$$

We also have the minimal expected delay under this scheme

$$D_{min} \geq \left( \frac{C}{d_{map}} + 2 \right) \cdot \frac{L}{B_{up}}$$



## Optimal Scheduling Problem

- Assume  $M$  CMs are active on an upstream channel with bandwidth  $B$ ; all frames are the same size  $L$ .
- Assume we know request arrivals from the  $i^{\text{th}}$  CM during a period of  $T$ , as  $A_i = \{a_{i,j} | 1 \leq j \leq J_i\}$ .
- Optimal Scheduling Algorithm generates  $n$  maps with size of  $t_k$ , so that the channel utilization  $\eta$  is maximized, where  $S_k = \{\text{scheduled requests pending during } t_k\}$ ,

$$\sum_{k=1}^n t_k = T \quad \text{and} \quad \eta = \frac{\sum_{k=1}^n |S_k| \cdot L}{T \cdot B}$$



## Approximate Algorithms

- Limitation of the Optimal Algorithm
  - High computational costs due to request dependency
  - Request arrival times are usually unknown *a priori*
- Approximate Algorithms (on-going work)
  - Greedy Algorithm
    - Requests are served in-order for a CM (FCFS)
    - Map is generated based on a delay threshold
  - Quota-based Greedy Scheduling
    - Each CM has a rate limit, e.g., 128 Kbps
  - Statistics Algorithm
    - Unique input characteristics of DOCSIS
    - Quasirandom inputs: each CM has at most one request outstanding



## Optimal DS Placement

### ➤ Upstream frame delay

- Request access delay: from CM issues a request to it recv the ACK
- Data access delay: from CM recv the ACK to CMTS recv the frame

### ➤ Previous Approaches

- mostly focused on request access delay
- few proposed AL's for reducing data access delay  
e.g., shortest-request-first, longest-request-first, priority-scheme

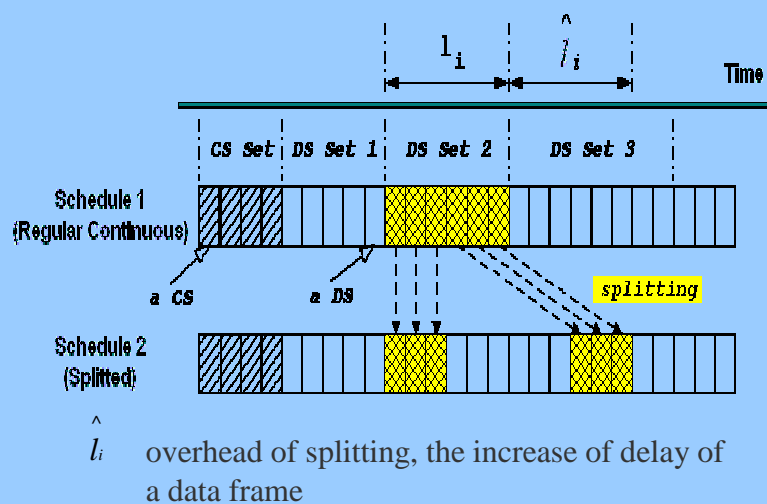
### ➤ Goal: Exploiting the request piggybacking feature

### ➤ Assumptions

- all requests in a single priority-class
- fixed map frequency/size, fixed ratio of CS's and DS's

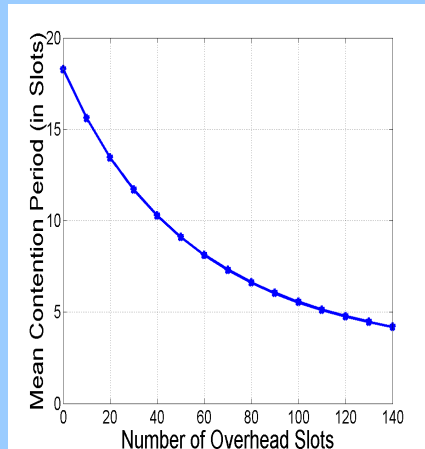


## Idea of Splitting DS's

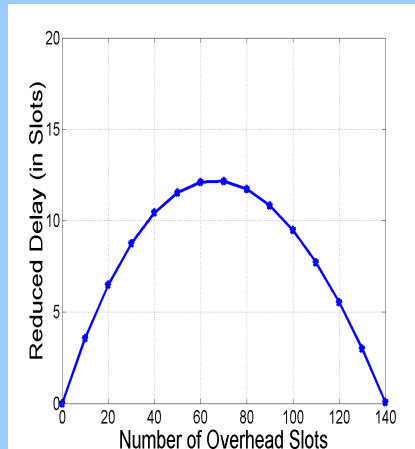




## Splitting Benefits v.s. Overheads



Mean Contention Period



Reduction in mean delay



## Optimization of Splitting Overhead

$$\hat{l}_i = \frac{\ln(C_{slot} \cdot \lambda \cdot U)}{\lambda \cdot U} - l_i$$

where  $C_{slot}$  is the mean contention delay and  $U=25$  us.

- $\hat{l}_i = 53$  when plugging the parameters from above
- **Limits: Simply splitting each DS set is not necessary**
  - Activate only when measured contention probability is high
  - Applying the approach only for high rate flows



## Summary and Current Work

---

---

### ➤ Summary

- Simple analysis of scheduling delay
- Optimal scheduling problem and approximate algorithms
- Optimal DS-placement

### ➤ Current Work

- Application-Aware Scheduling
  - Delay-sensitive: gaming, voice stream
  - BW-sensitive: video phone, VOD



---

---

*Thanks for your attention!*