



Split Transaction Budgeting Algorithm

John S. Howard
Engineering Manager
Intel Corporation



Budgeting Algorithm Outline



- ▶ **Terms**
- ▶ **General Algorithm Comments**
- ▶ **Non-split Transactions**
- ▶ **Budgeting Split Transactions**
 - Core Spec Requirements
 - EHCI Requirements
 - Algorithm implementation tradeoffs

- ▶ **Information in:**
 - Core Spec Chapters 5, 8 & 11
 - EHCI Spec.



Budget and Schedule

- ▶ **Host determines when transactions run**
 - The “Budget”: N (micro-)frames of information
 - Host builds description of transactions requested to run
- ▶ **Host Controller issues transactions**
 - The “Schedule”: $M \times N$ (micro-)frames of transactions
- ▶ **Budget and schedule must match**
 - Both are HC dependent
- ▶ **TT behavior must correspond**
 - Specified by core USB spec
- ▶ **Mismatched budget/schedule/TT causes errors**
- ▶ **Budgeting only applies to periodic transactions**



Repeating Frame Patterns

Budget



- ▶ **Budget consists of a repeating pattern of frames**
 - Likely a 2^N number of frames
 - Likely 64 or less frames for reasonable space usage
 - Example algorithm uses 8 frames
- ▶ **Schedule repeats budget pattern “endlessly”**
 - Schedule can be same number of frames as budget



Budgeting Algorithms

- ▶ Many different algorithms possible
- ▶ Any algorithm has requirements and constraints
 - Determined by USB core spec. and HC specification(s)
- ▶ Simple budget algorithm in use for classic USB
- ▶ Split transactions require new budget algorithm
- ▶ This session illustrates one budget algorithm
 - Focus is on split transaction budget



Non-Split Budgeting

- ▶ **Classic and high-speed bus allocations easy**
 - **Count bytes for transaction in (micro-)frame**
 - **Well known for classic**
 - ◆ **90% / 10% for periodic/non-periodic frame**
 - ◆ **Worst case (i.e. include bit-stuffing)**
 - ◆ **Transaction overheads well known (chap 5)**
 - **Similar for high-speed**
 - ◆ **80% / 20% for microframe**
 - ◆ **Worst case (i.e. include bit-stuffing)**
 - ◆ **New transaction overheads (chap 5)**



Transaction Overhead

- ▶ **Bus time requirements of a transaction includes:**
 - **Token packet (Sync + 3 bytes + EOP)**
 - ◆ 4 bytes for split token vs. 3 for non-split token
 - **Data packet (Sync + 3 bytes + wMaxPacketSize + EOP)**
 - ◆ If present
 - **Handshake packet (Sync + 1 byte + EOP)**
 - ◆ If present
 - **2 Bus turnaround times**
 - **Think_time (Inter transaction gap)**
 - ◆ HC dependent



Overheads Reference

▶ Classic overheads (FS byte times, T+D+H):

- FS_Isoch 9
- FS_Interrupt 13
- LS_Interrupt 19*8

▶ High-speed split overheads (HS byte times):

- Token_Same_direction 39
- Token_Change_direction 29
- Data_Same_direction 19
- Data_Change_direction 9
- Handshake_only 7

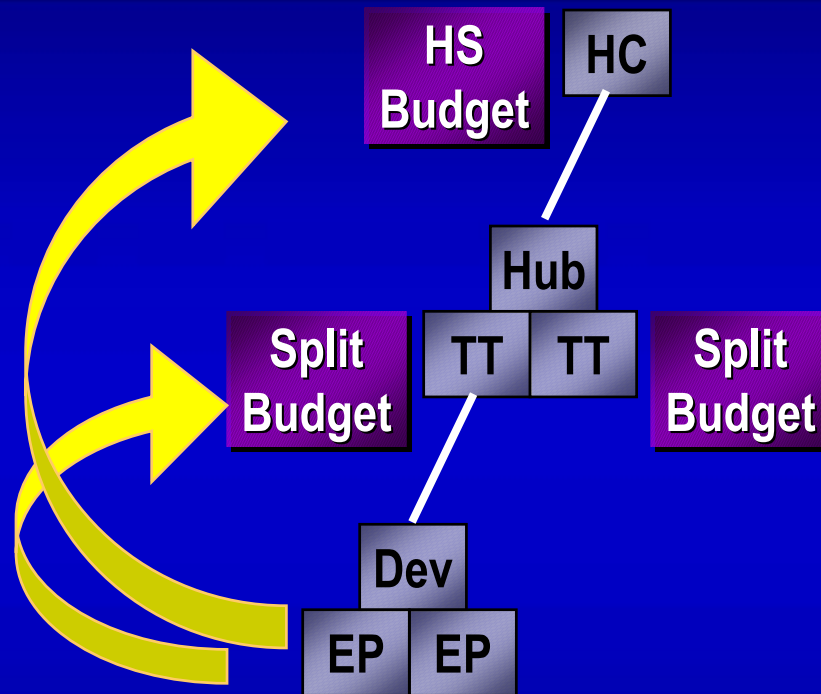


TTs and Split Budgeting

- ▶ **One or more Host Controllers per system**
 - One High-speed budget per HS Host Controller
 - One classic budget per USB1.1 Host Controller
- ▶ **One Bus per Host Controller**
- ▶ **One or more Hubs per bus**
- ▶ **One or more TTs per Hub**
- ▶ **A classic bus per TT**
 - One split budget per TT
- ▶ **Split budgeter allocates HS and classic bus times**



HC/TT/Device_endpoint

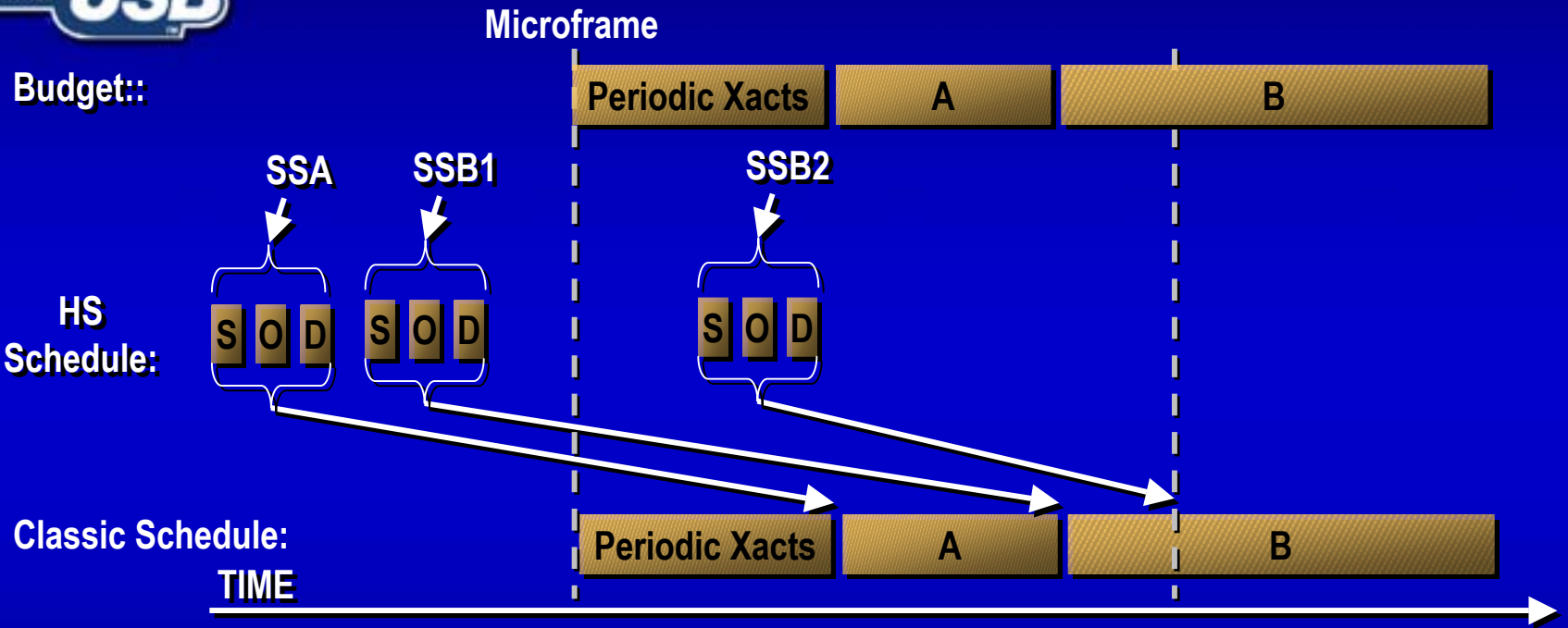


▶ New configured EP allocates 2 bus times:

- Classic from split_budget of “parent” TT
 - ◆ To determine in what microframes classic transaction can run
- High-speed from HC HS budget, per microframe
 - ◆ For split transactions on HS bus

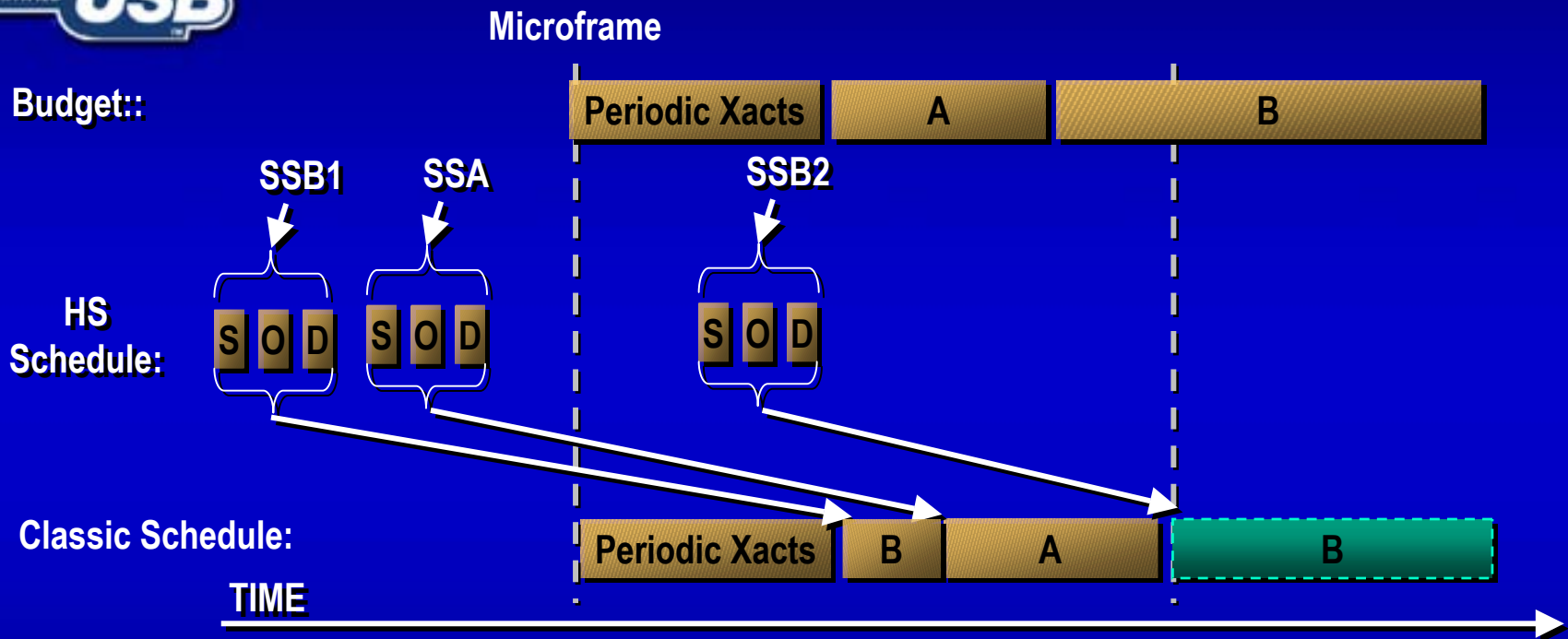


Budget/Schedule Example



- ▶ Two isoch OUT endpoints: A, B (spans uframe)
- ▶ Does A/B split-start order matter?

Budget/Schedule Example (Error)



► Does A/B split-start order matter?

– YES!!

– TT sees classic B and HS SSB2 as errors



Core Spec Requirements

▶ Allocate bus times

- HS split transactions with bit-stuffing, 80% x 7500 bytes max
- Classic without bit-stuffing, 90% x 6/7 x 1500 bytes max

▶ Only 2^N periods allowed, (new) periodic isochronous

▶ Split Processing:

- Determine start- & complete- split patterns
 - ◆ Fewer CSs at end of frame (chap 11 for details)
- At most 16 SS per microframe

▶ Split Ordering

- CSs for all endpoints in each microframe matches SS order
- SSs & CSs scheduled in budget order for microframe

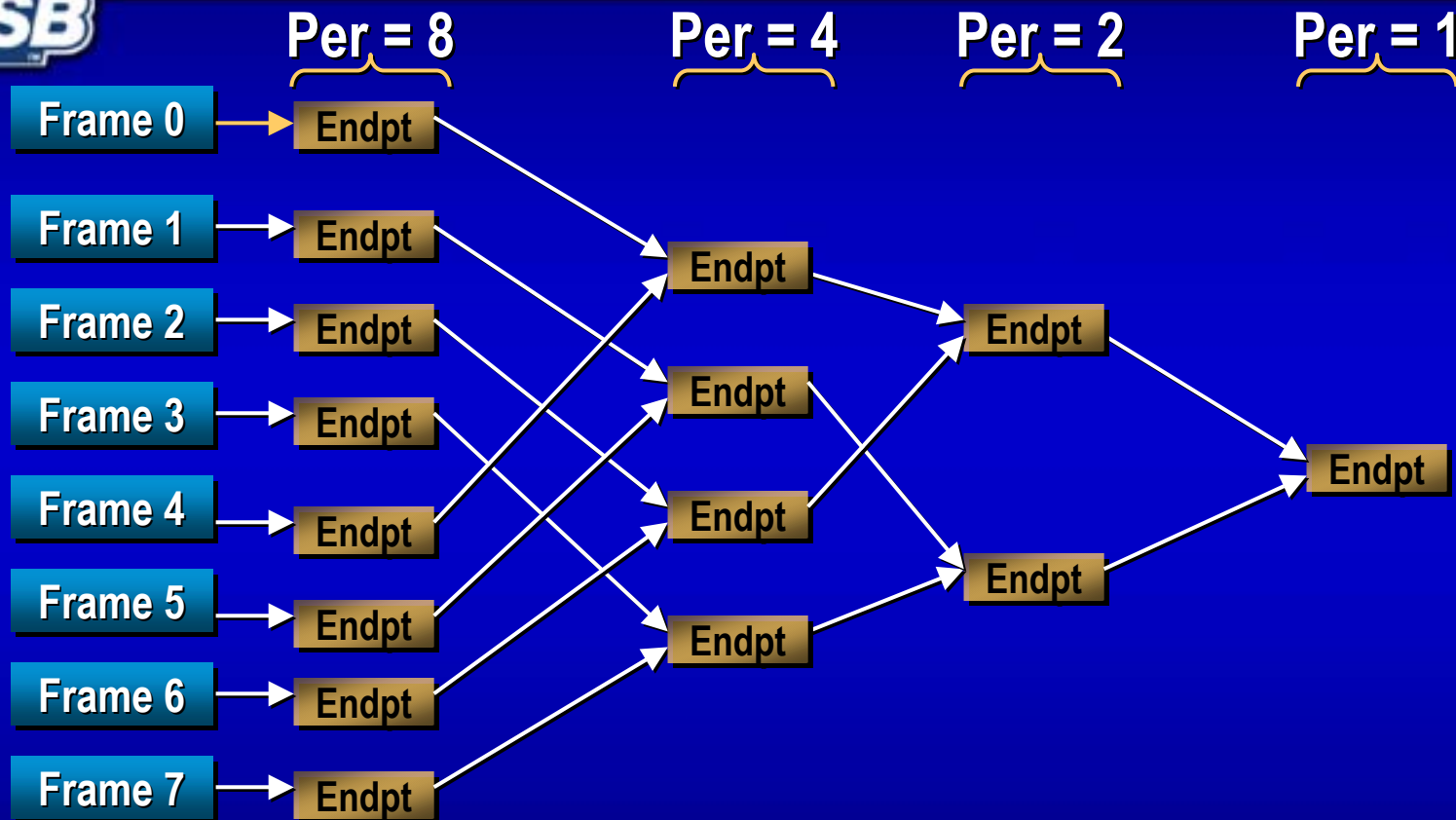


EHCI Requirements

- ▶ **Schedule “large” isoch first in frame**
 - Large is $> 1157 \times \frac{1}{2}$, i.e. 579 bytes
 - Not only `wMaxPacketSize`,...
 - Must include transaction overhead also
- ▶ **Interrupt in decreasing period order**
 - Dictated by Qhead endpoint “tree”



Endpoint Trees



- ▶ One struct per allocated endpoint
- ▶ Linked into correct period frame list(s)



Algorithm Tradeoffs

- ▶ **Minimize changes to current budget/schedule**
 - Adding new endpoint should avoid change to others
- ▶ **Desirable to balance allocation in all frames**
 - Find least allocated frame(s) for new allocation
 - ◆ First fully allocated frame precludes more period 1 endpoints
- ▶ **What order for endpoints with same period?**
 - Chose “oldest to newest” order toward end of frame
- ▶ **All isochronous endpoints, then all interrupt**
 - Other sequences are possible to implement

Split Endpoint Order in Frame



Split Isoch. endpoints

Decreasing period



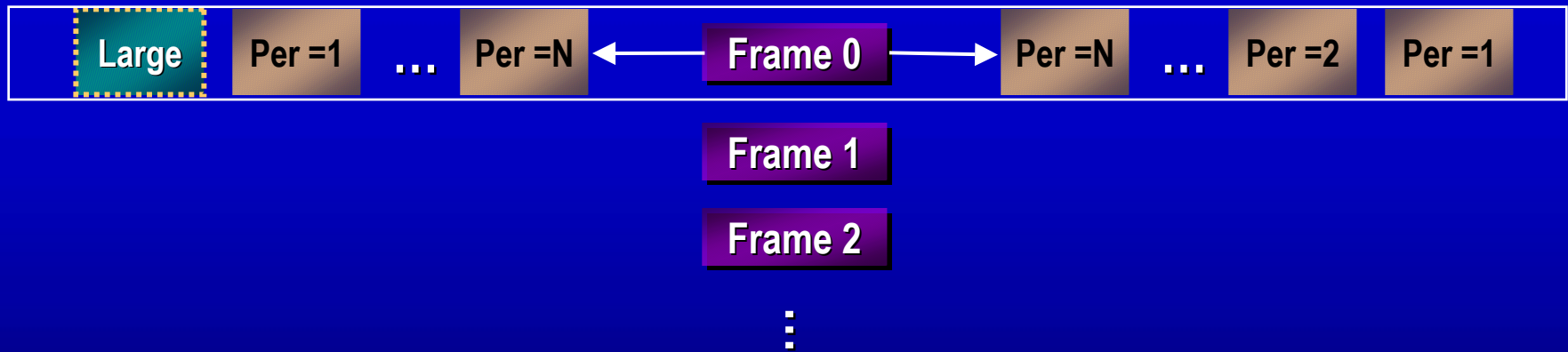
same period
old → new

Split Interrupt endpoints

Decreasing period



same period
old → new



- Only one large isoch allowed in any frame



Algorithm Comments

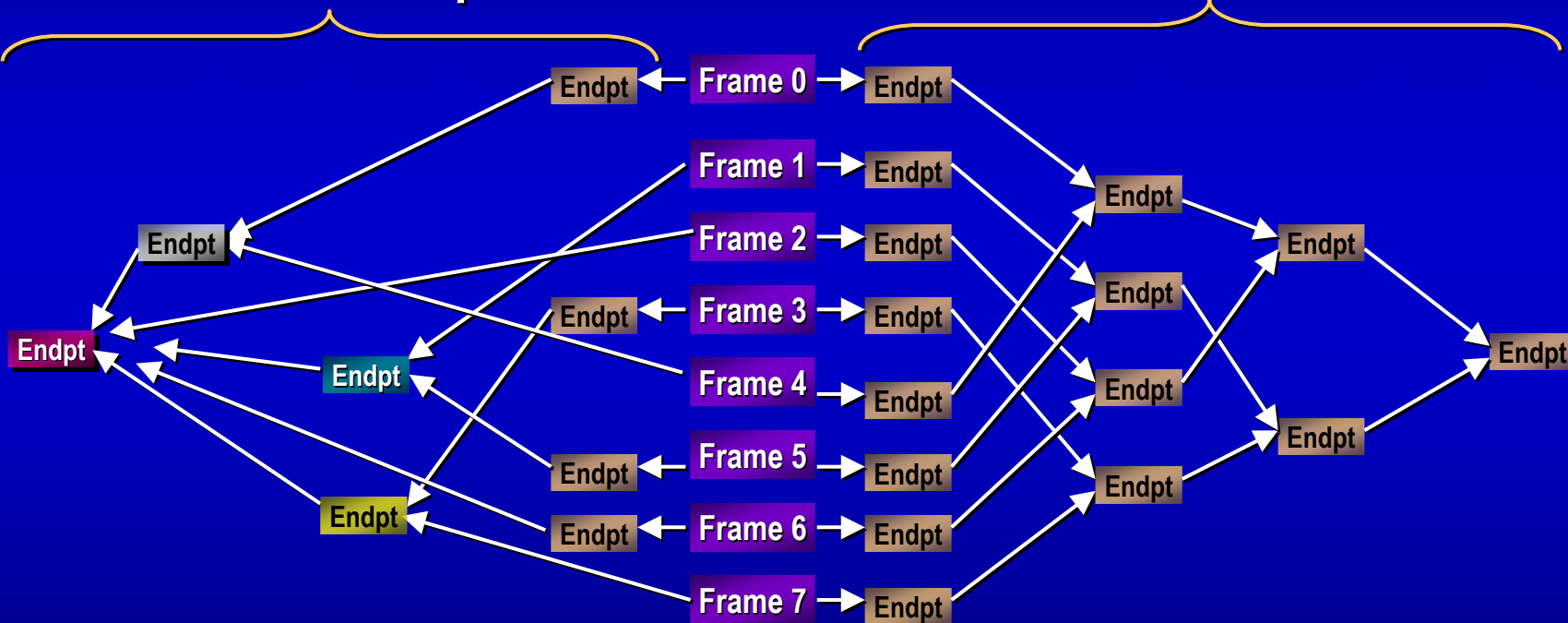
- ▶ **Preserve old to new order within same period**
 - Least impact on other allocated endpoints
 - Most isoch. will be period = 1
 - Most isoch. allocations at end of isoch. portion of frame
- ▶ **Isoch. order in frame is preferred**
 - Allows compact budget, no budget “holes”
 - But, EHCI dictates opposite order for interrupts



Budget Data Structure

Isochronous Endpoints

Interrupt Endpoints



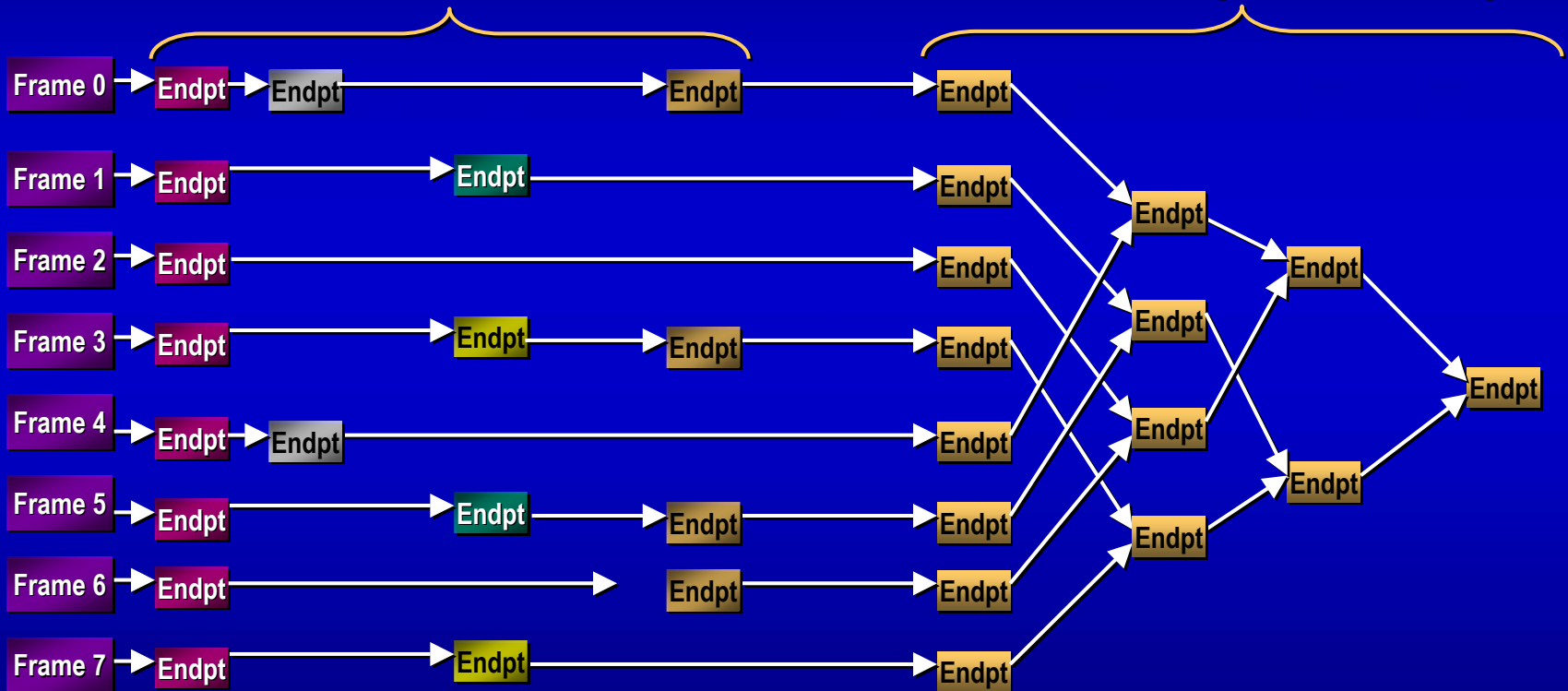
► Budget uses endpoint tree to track split isoch & interrupt



Schedule Data Structure

Ordered Isochronous siTDS

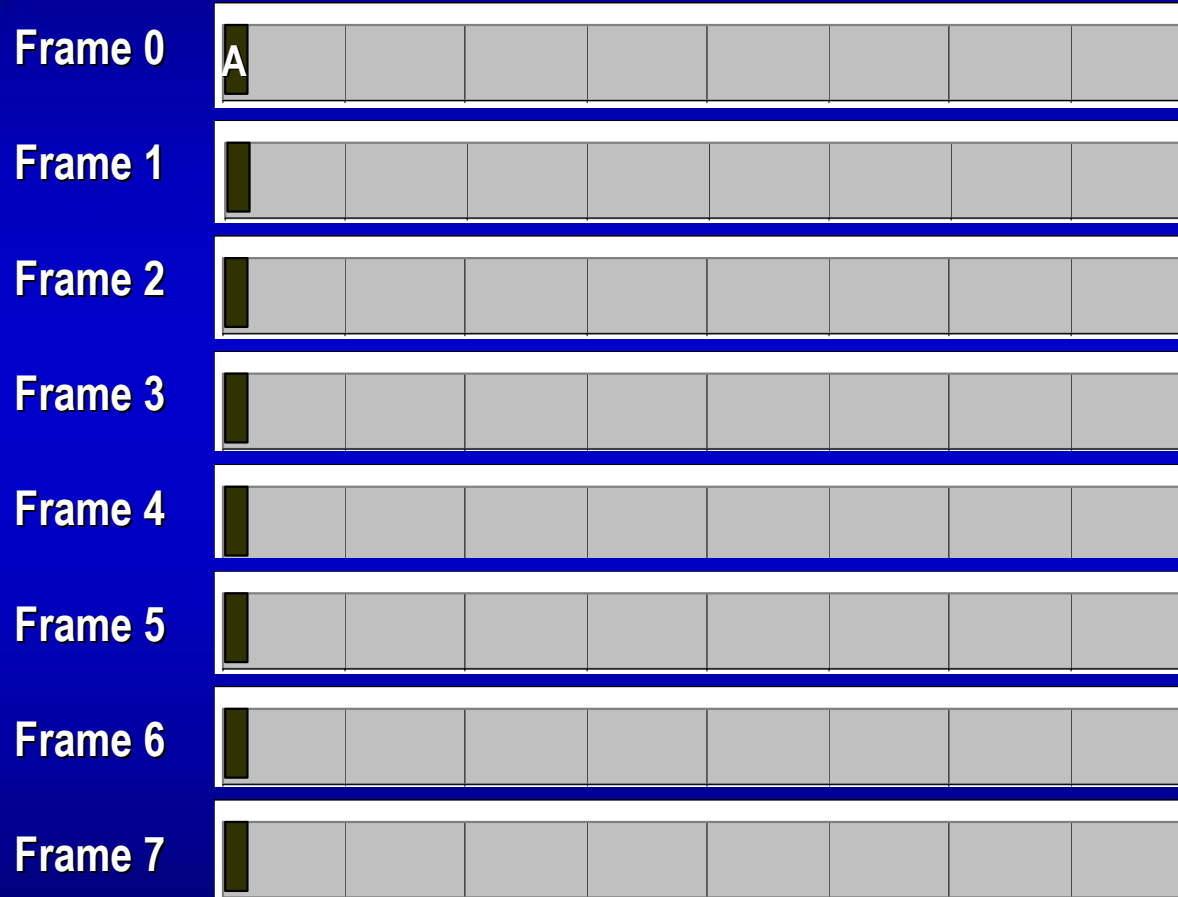
Ordered Interrupt Qheads/qTDs



- ▶ Schedule uses interrupt Tree, but replicates isoch. TDs
- ▶ HS nonsplit TDs are unordered and can be “anywhere”



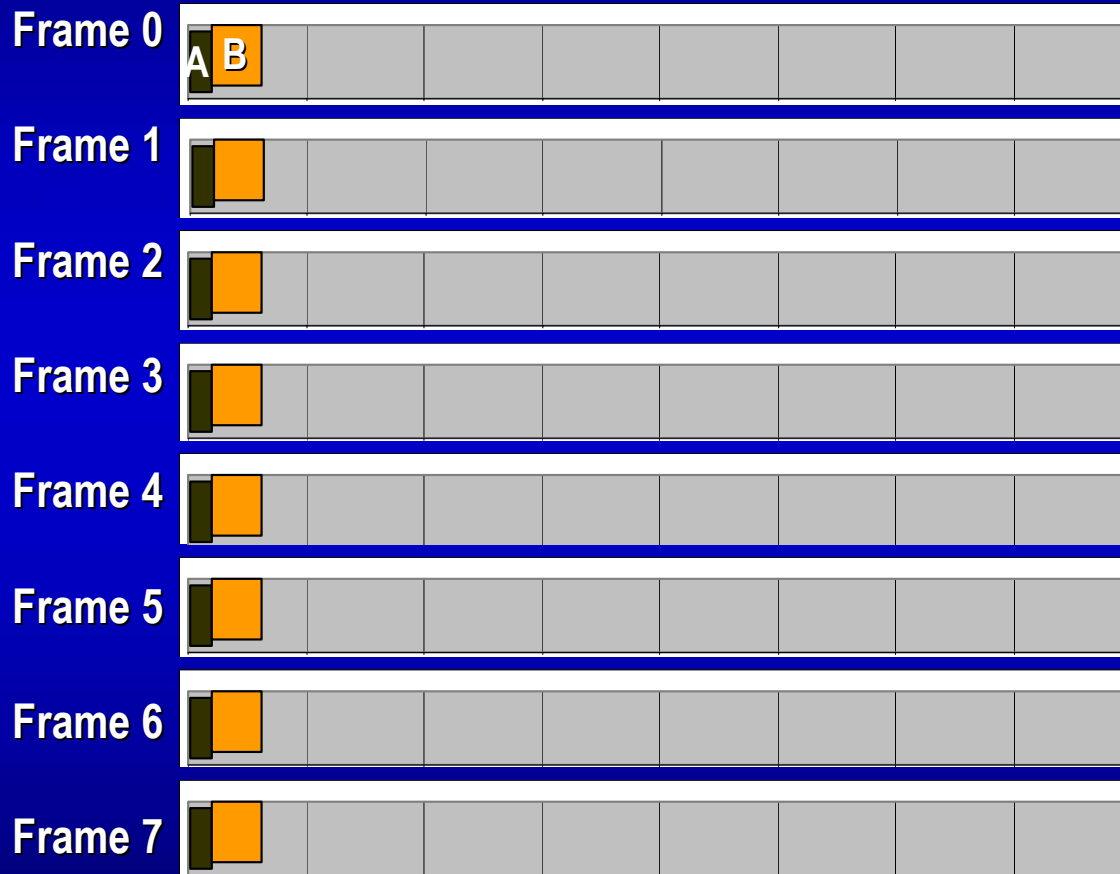
Example Budget - 1



► A: Isoch out, wMaxPacketSize=19, Period=1



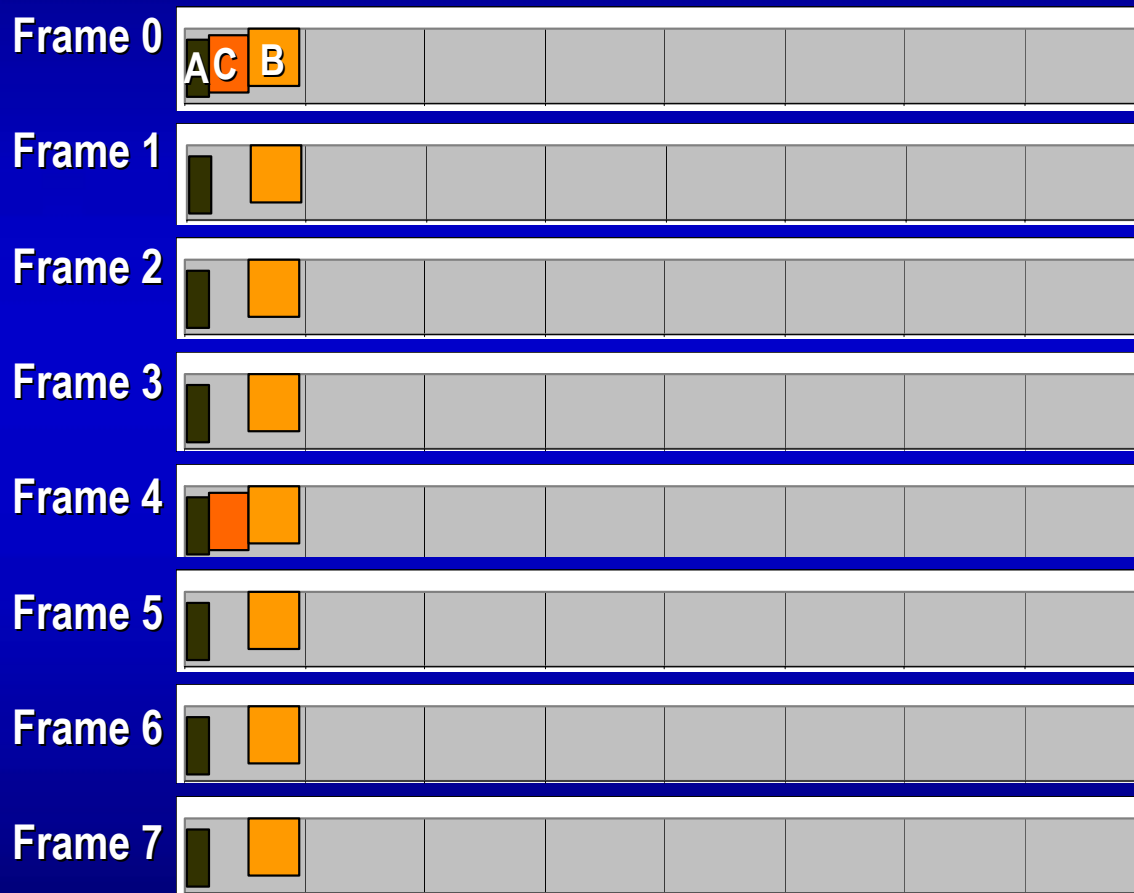
Example Budget - 2



► B: Inter out, wMaxPacketSize=64, Period= 1



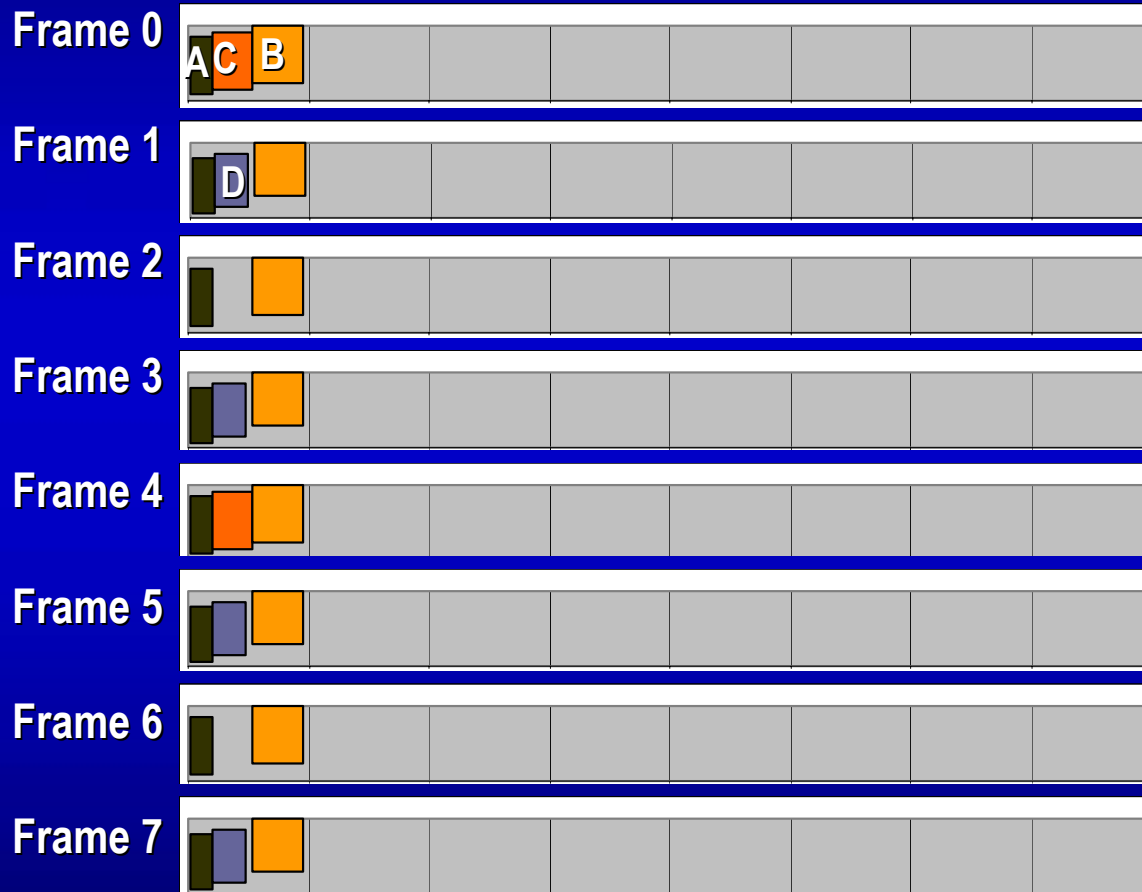
Example Budget - 3



► C: Inter in, wMaxPacketSize=45, Period= 4



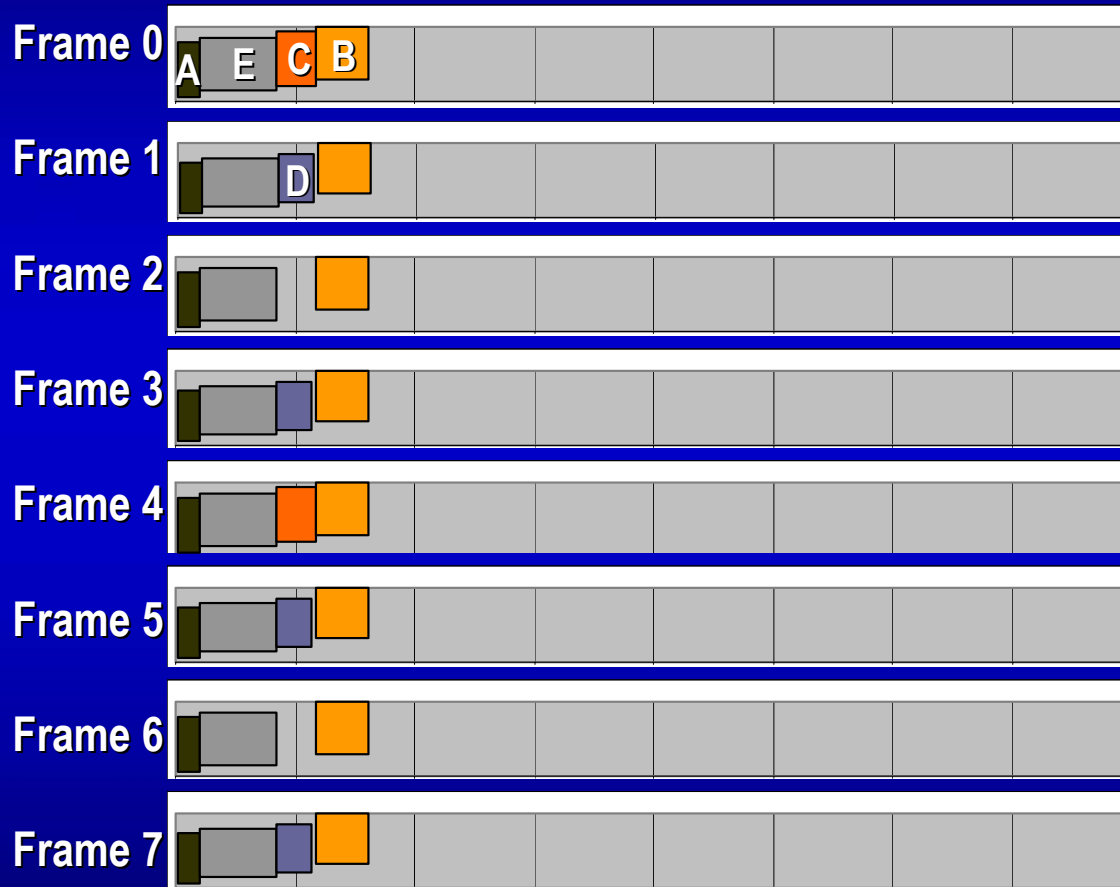
Example Budget - 4



► D: Inter in, wMaxPacketSize=36, Period= 2



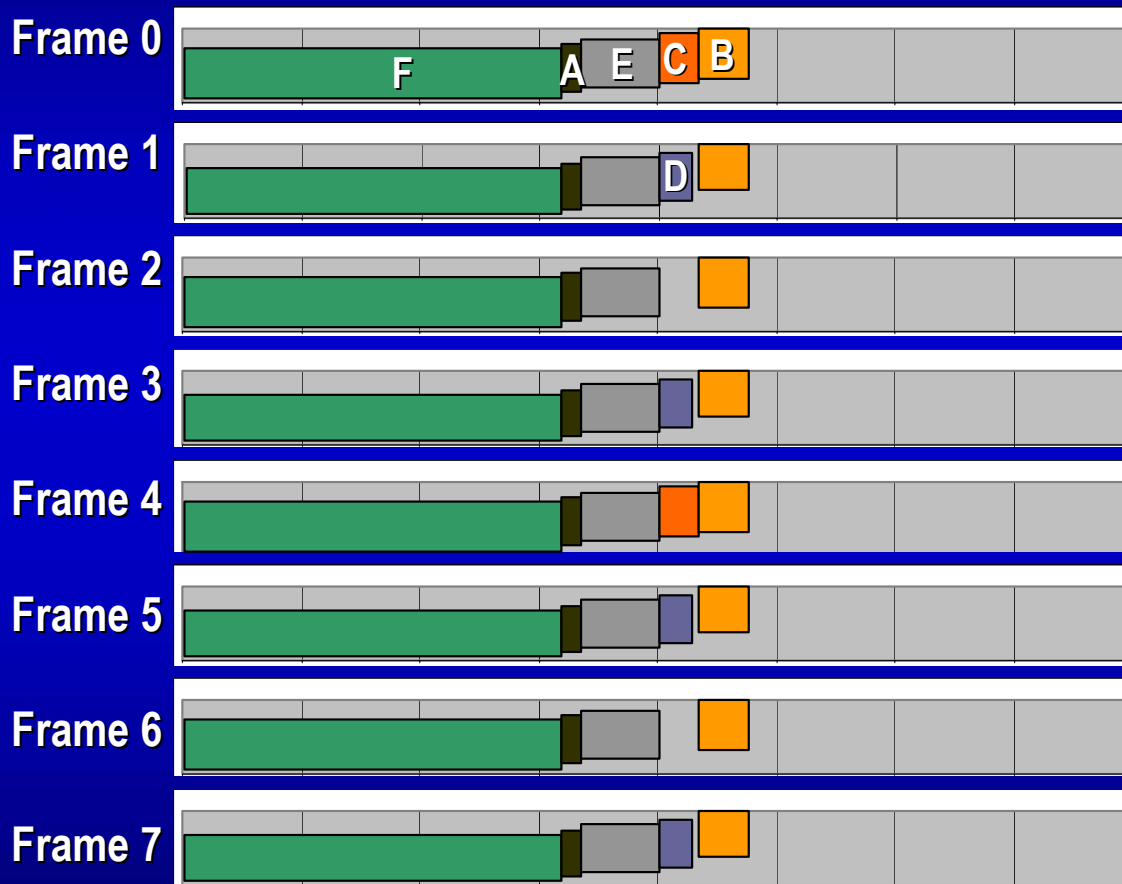
Example Budget - 5



► E: Isoc in, wMaxPacketSize=110, Period= 1



Example Budget - 6



► F: Isoch in, wMaxPacketSize=580, Period= 1



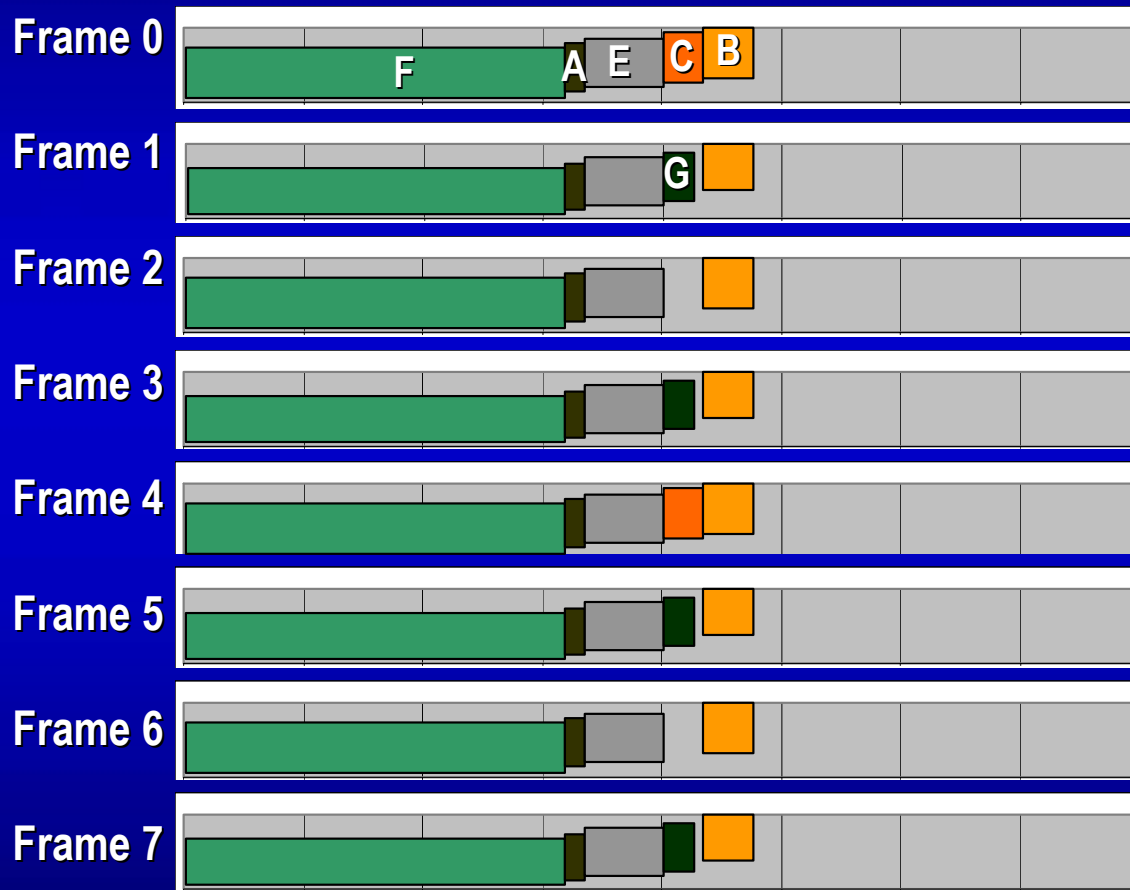
Example Budget - 7



► **G: Inter in, wMaxPacketSize=32, Period= 2**



Example Budget Deallocation



Deallocation D: B & G “compacted”

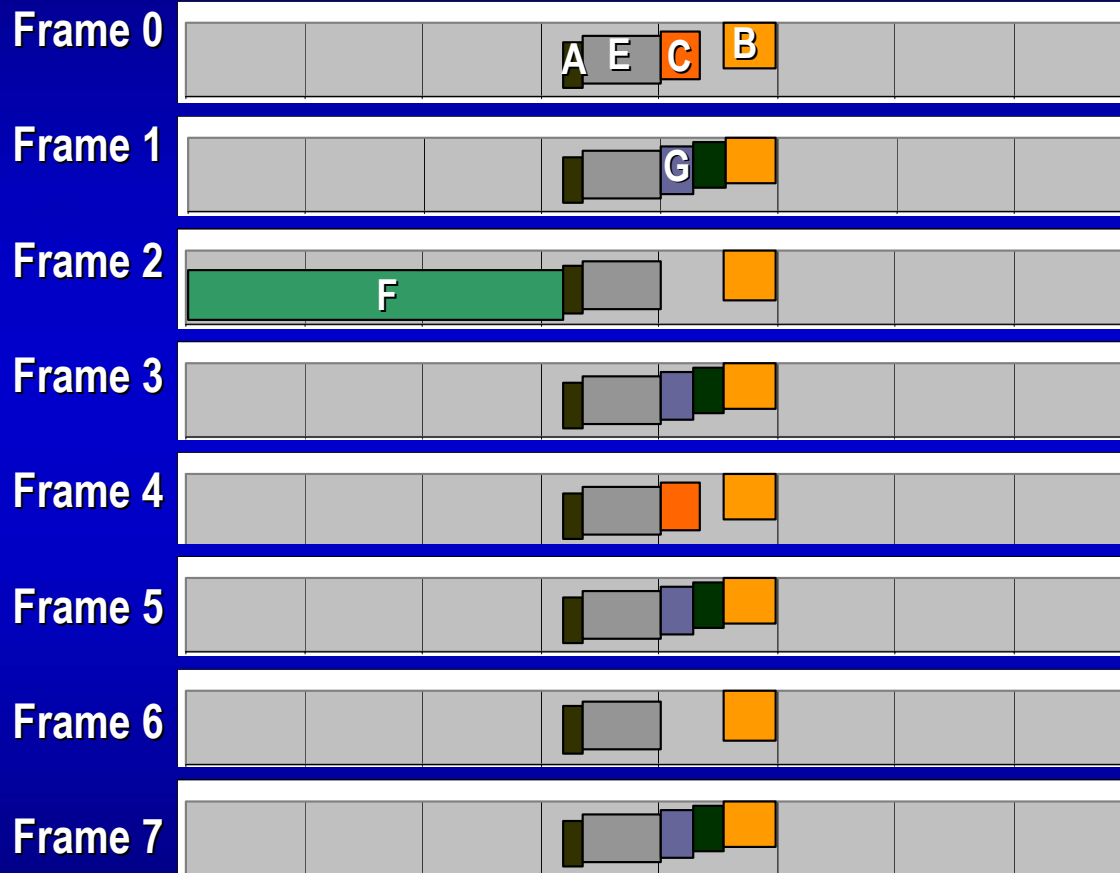
Calculated Budget Parameters



<i>ID</i>	<i>Microframe</i>	<i>StartSplits</i>	<i>CompleteSplits</i>
<i>A</i>	<i>2</i>	<i>1</i>	<i>0</i>
<i>B</i>	<i>3</i>	<i>1</i>	<i>3</i>
<i>C</i>	<i>3</i>	<i>1</i>	<i>3</i>
<i>E</i>	<i>2</i>	<i>1</i>	<i>4</i>
<i>F</i>	<i>-1</i>	<i>1</i>	<i>6</i>
<i>G</i>	<i>3</i>	<i>1</i>	<i>3</i>



Example Periodic Isoch



- ▶ Creates “hole” that can be reclaimed by bulk/control
 - Or other large periodic isoch allocations



Budgeting Summary

- ▶ **Classic budgeting is well understood**
- ▶ **High-speed budgeting is a derivative of classic**
- ▶ **Split transaction budgeting is new and complex**
- ▶ **Split budgeting must be done correctly**
- ▶ **A split budgeting algorithm exists and works**
- ▶ **Contact: techsup@usb.org**