Analysis of a Simple Approach to Modeling Performance for Streaming Data Applications

Jonathan C. Beard
Roger D. Chamberlain

SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St.Louis

Work supported by:

NSF exegy

August 2013

Outline

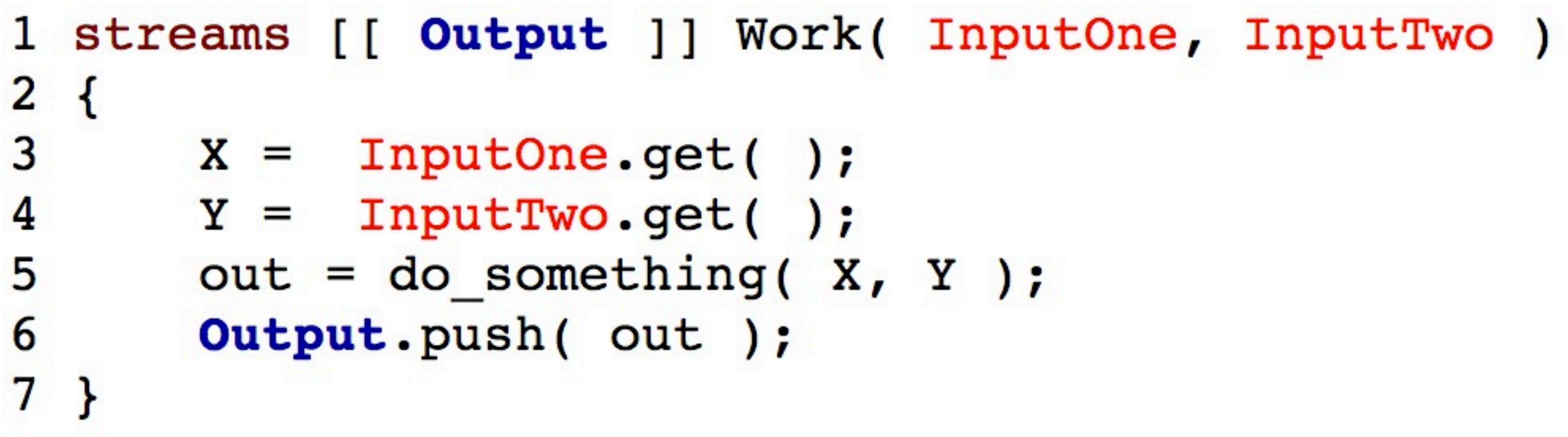
- We introduce a **simple model** to **estimate throughput** and **inform buffering capacity**

- The model is **tailored** to **stream processing**

- Is applicable to applications deployed on **heterogeneous architectures**

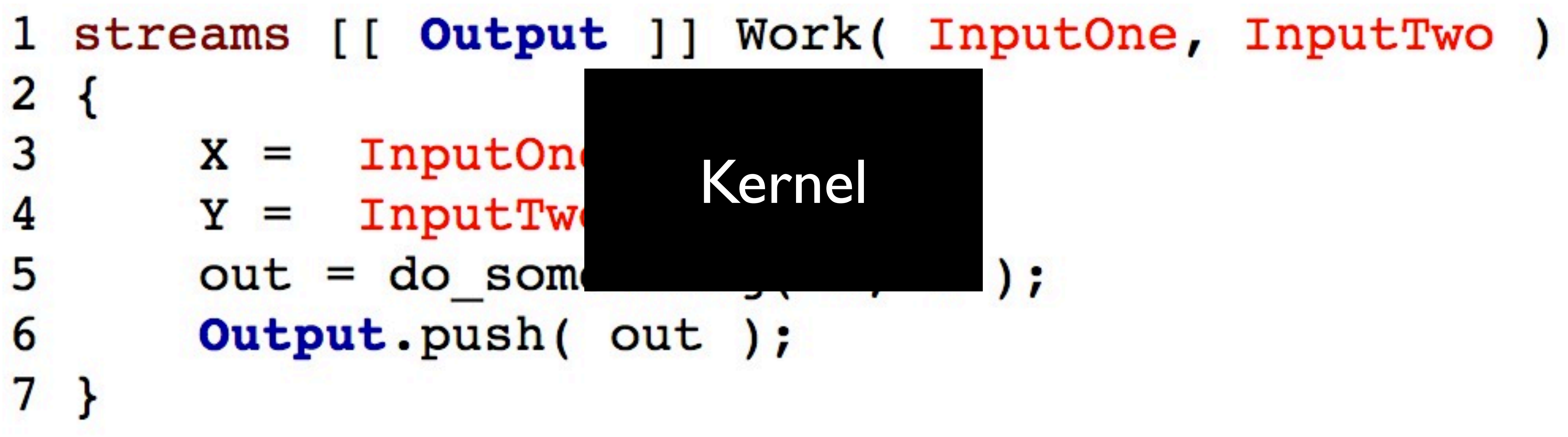
- We **empirically evaluate** the proposed model and discuss instances **where it works** and **where it might not**

Stream Processing Intro - Kernel



```
streams [[ Output ]] Work( InputOne, InputTwo )
{
        X =  InputOne.get( );
        Y =  InputTwo.get( );
        out = do_something( X, Y );
        Output.push( out );
}
```

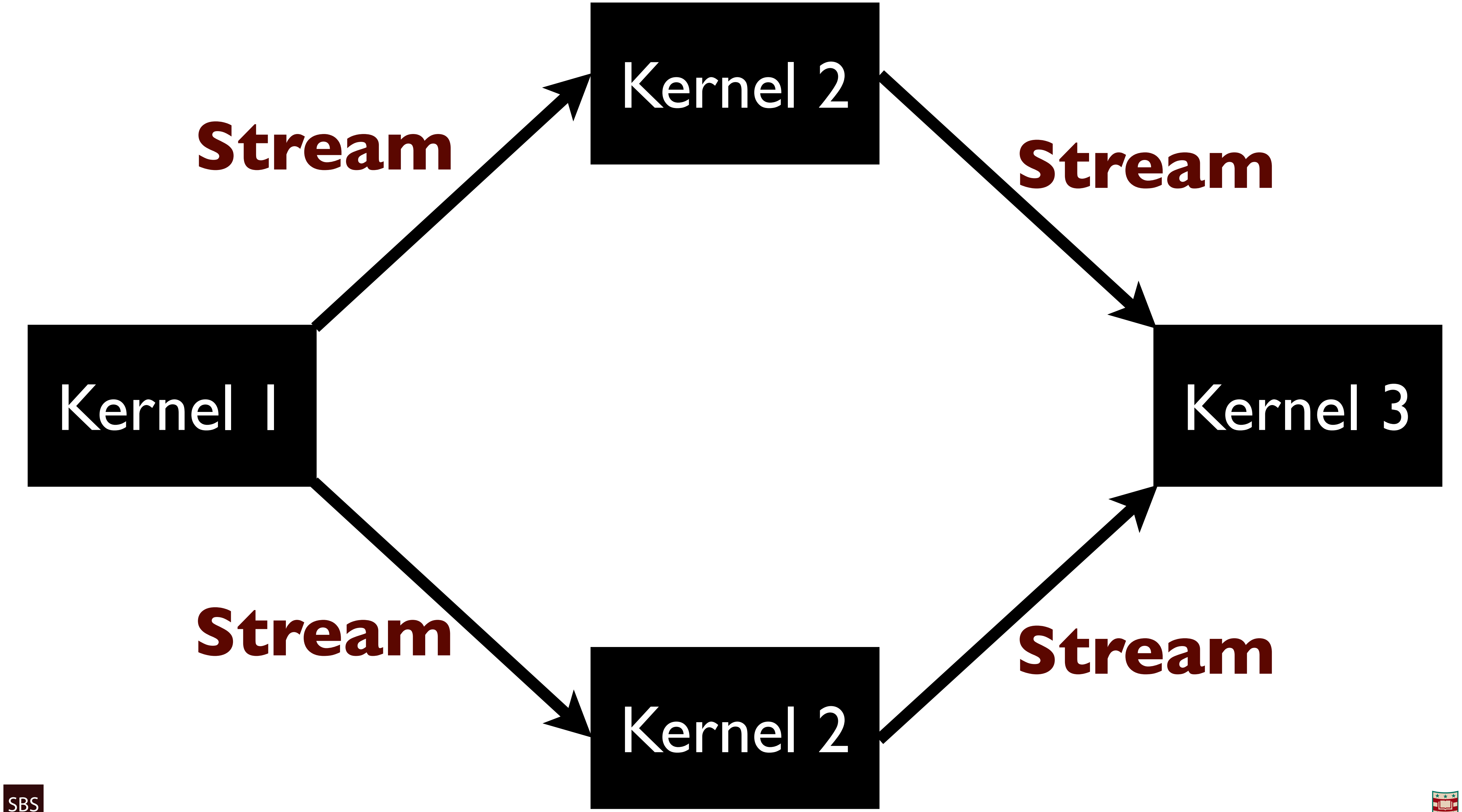
Stream Processing Intro - Kernel



```
streams [[ Output ]] Work( InputOne, InputTwo )
{
      X =  InputOn
      Y =  InputTw
      out = do_som                      );
      Output.push( out );
}
```

Kernel

Stream Processing Intro - Streams

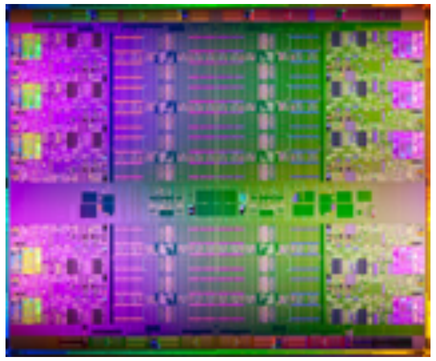


Stream Processing Intro - Languages

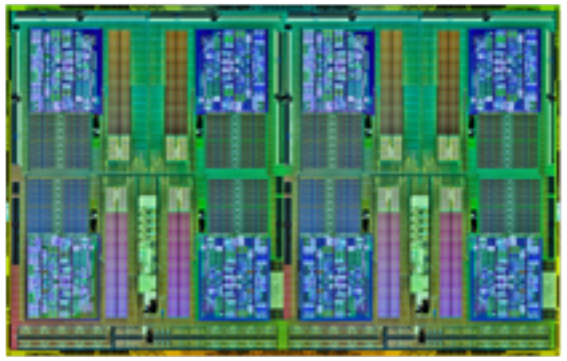
- Academic Systems: Auto-Pipe, Brook, Cg, S-Net, StreamIt, and Streams-C
- Commercial Systems: Impulse C and IBM's System S

Stream Processing Intro - Mapping I

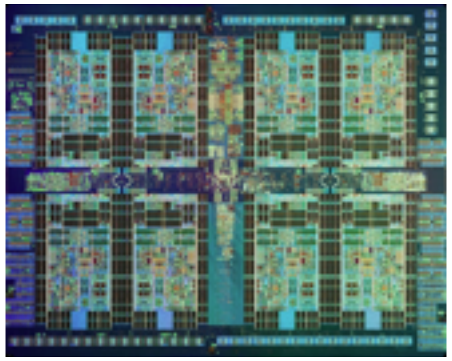
Multicore chips



Intel Xeon E7 (10-core)

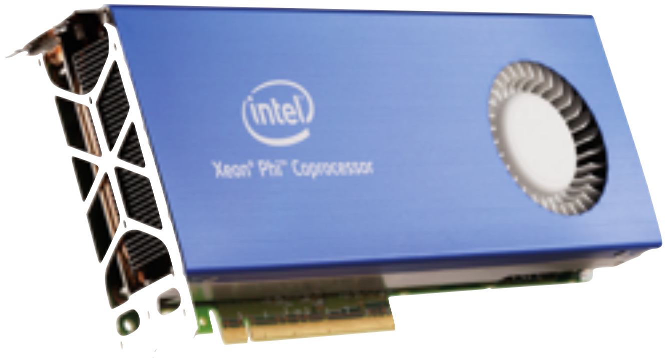


AMD Opteron 6300 (16-core)



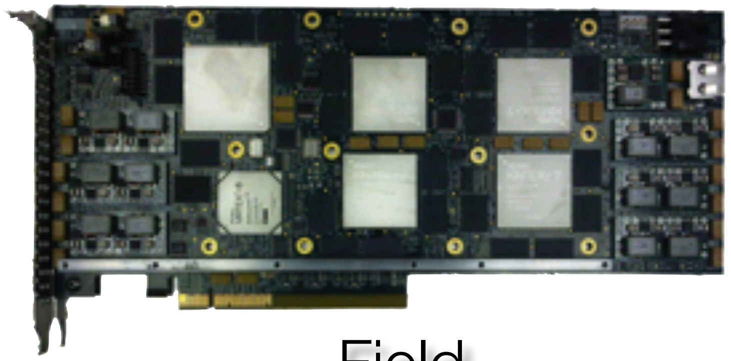
IBM Power7 (8-core)

Specialized Co-Processors



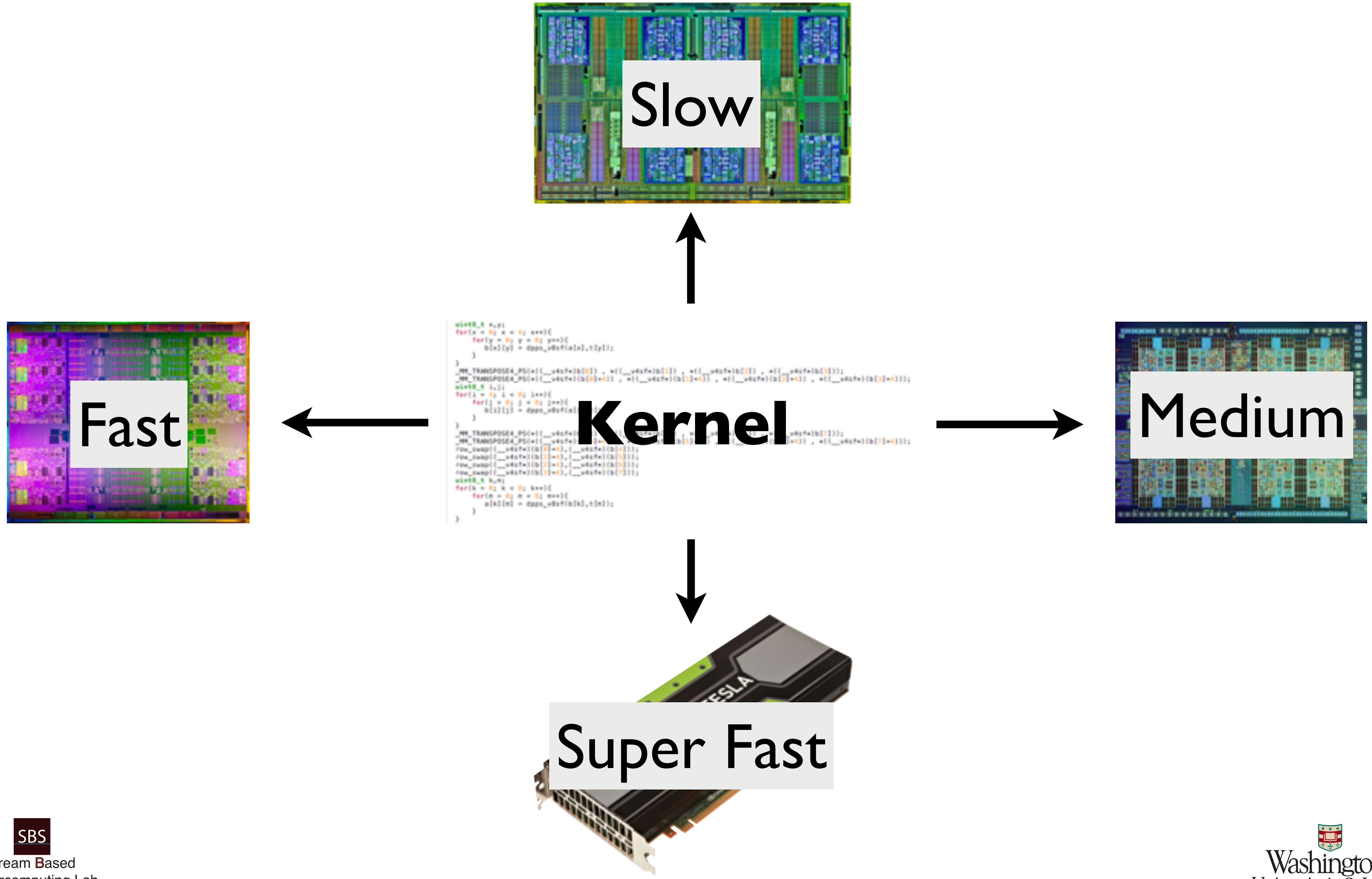
Intel Phi (61-core)

General Purpose Graphics Processor
(GPGPU)



Field
Programmable
Gate Array
(FPGA)

How does our kernel perform on each compute resource?



Slow

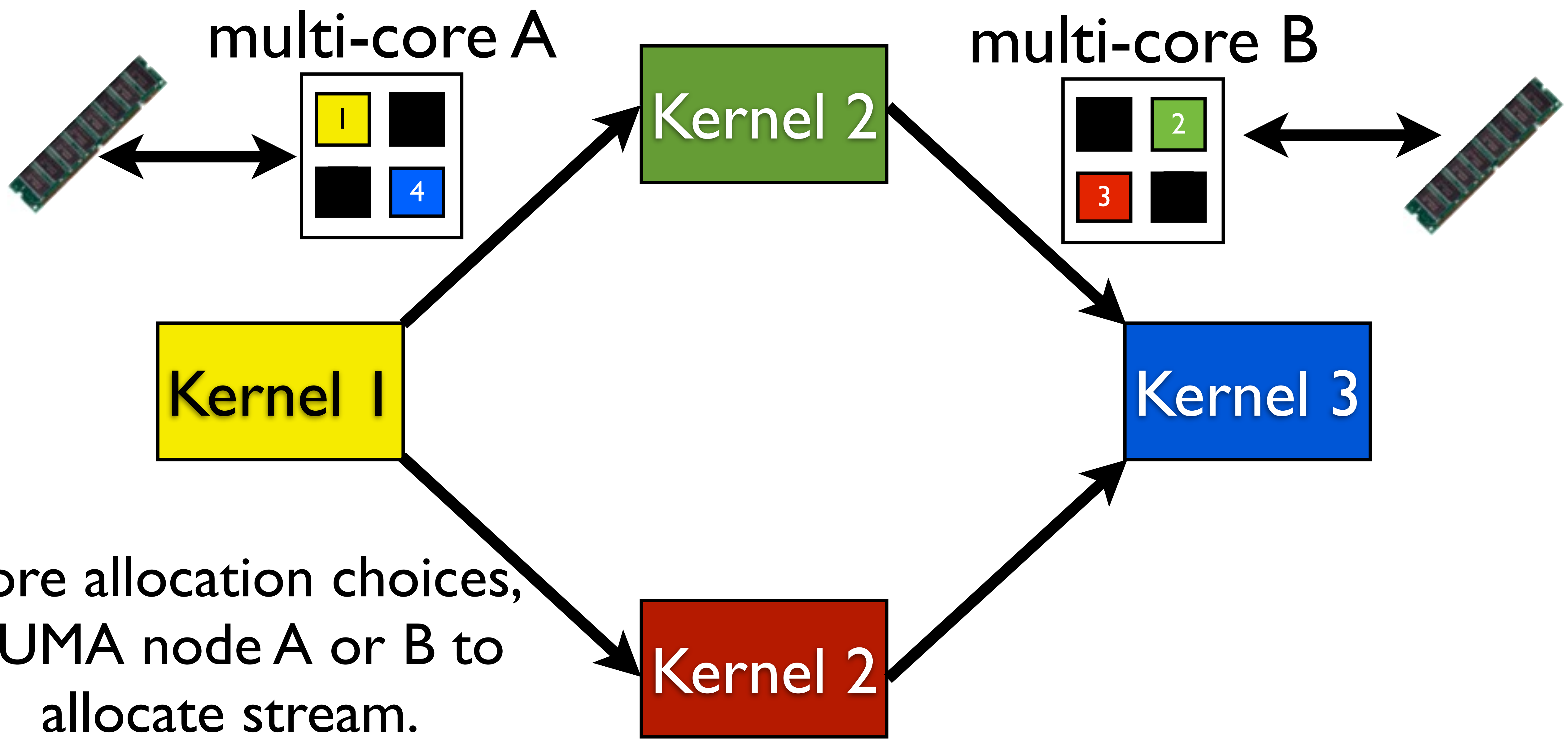
Fast

Kernel

Medium

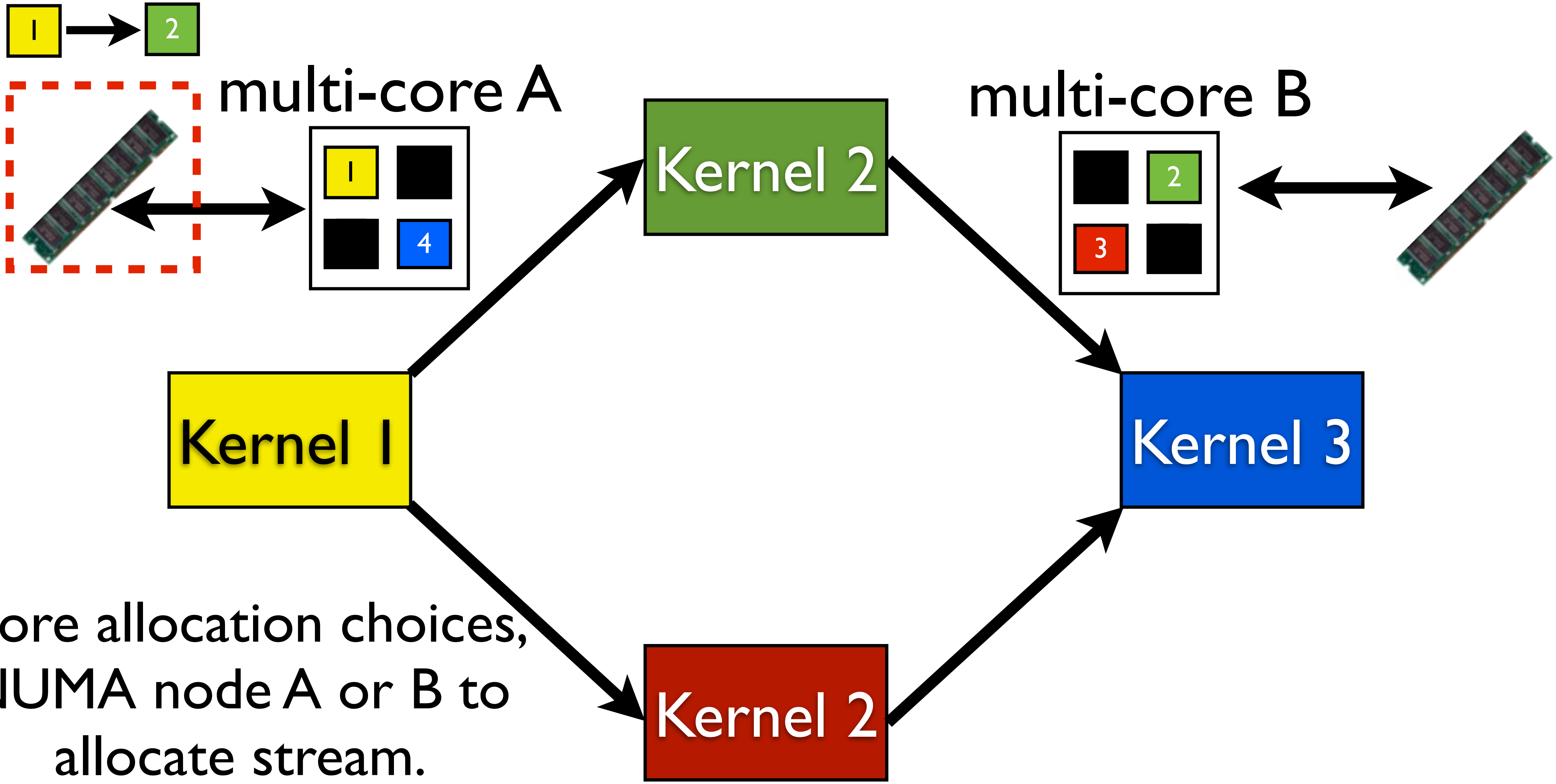
Super Fast

Stream Processing Intro - Mapping 3



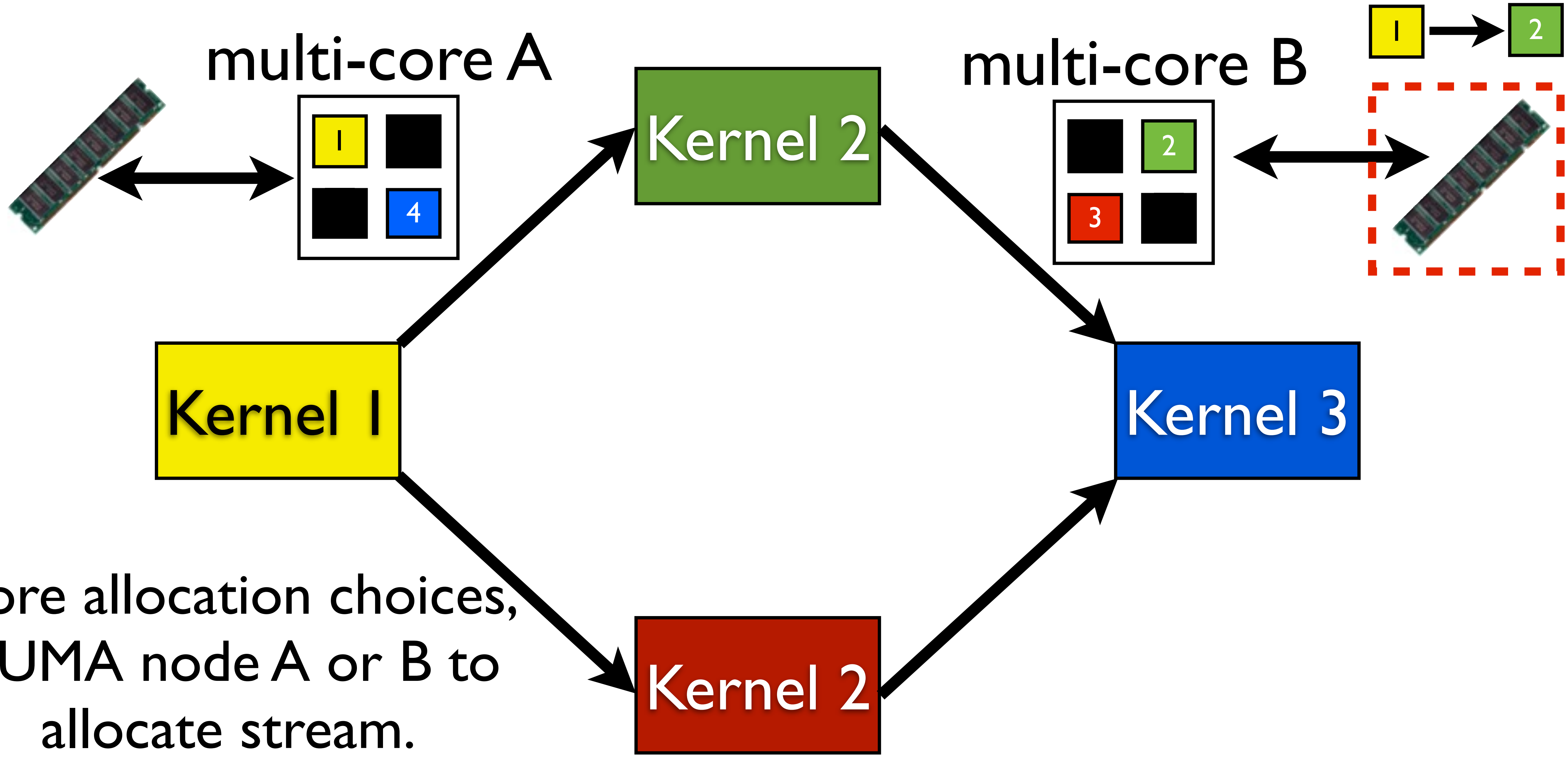
More allocation choices, NUMA node A or B to allocate stream.

Stream Processing Intro - Mapping 3



More allocation choices, NUMA node A or B to allocate stream.

Stream Processing Intro - Mapping 3

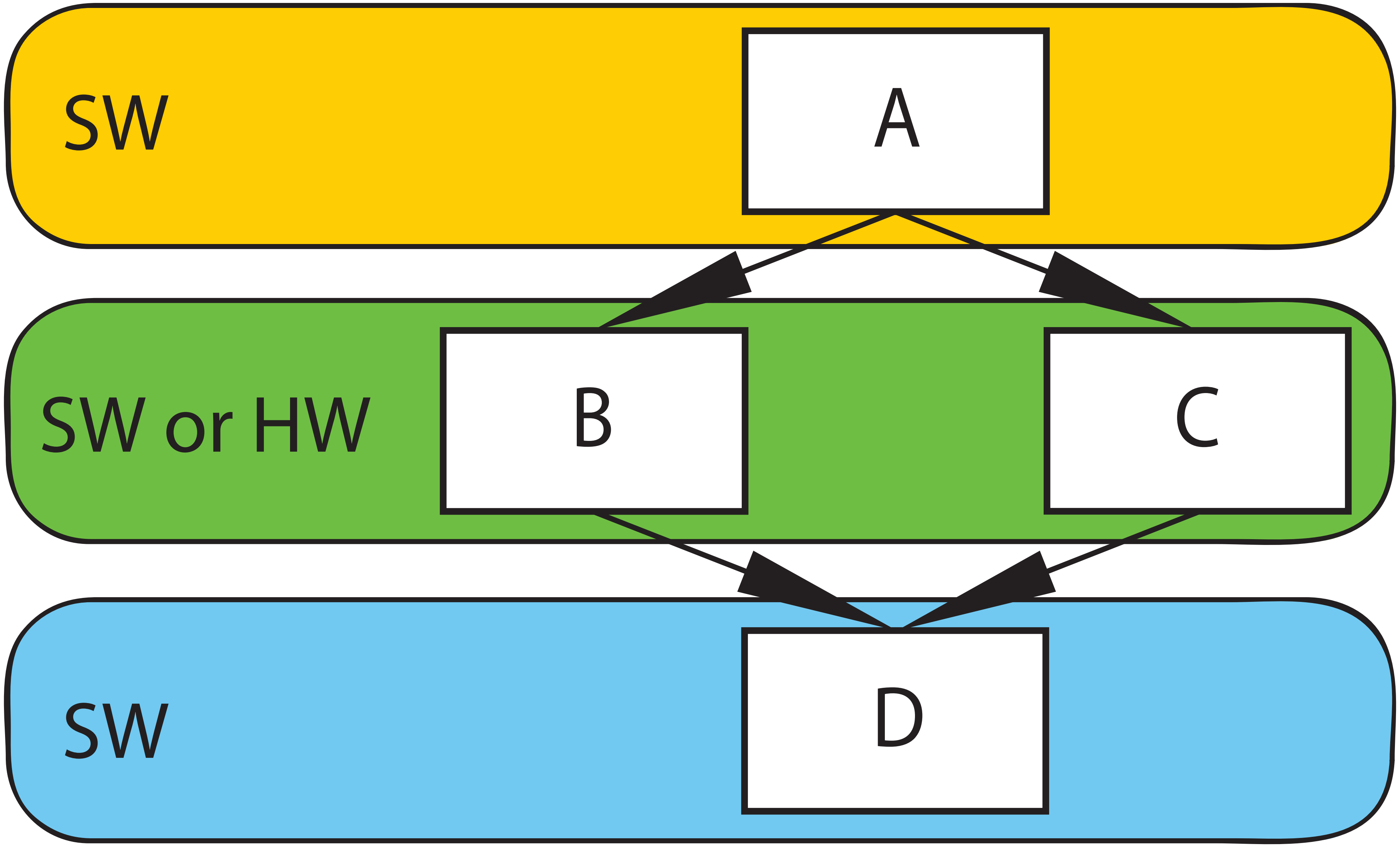


More allocation choices, NUMA node A or B to allocate stream.

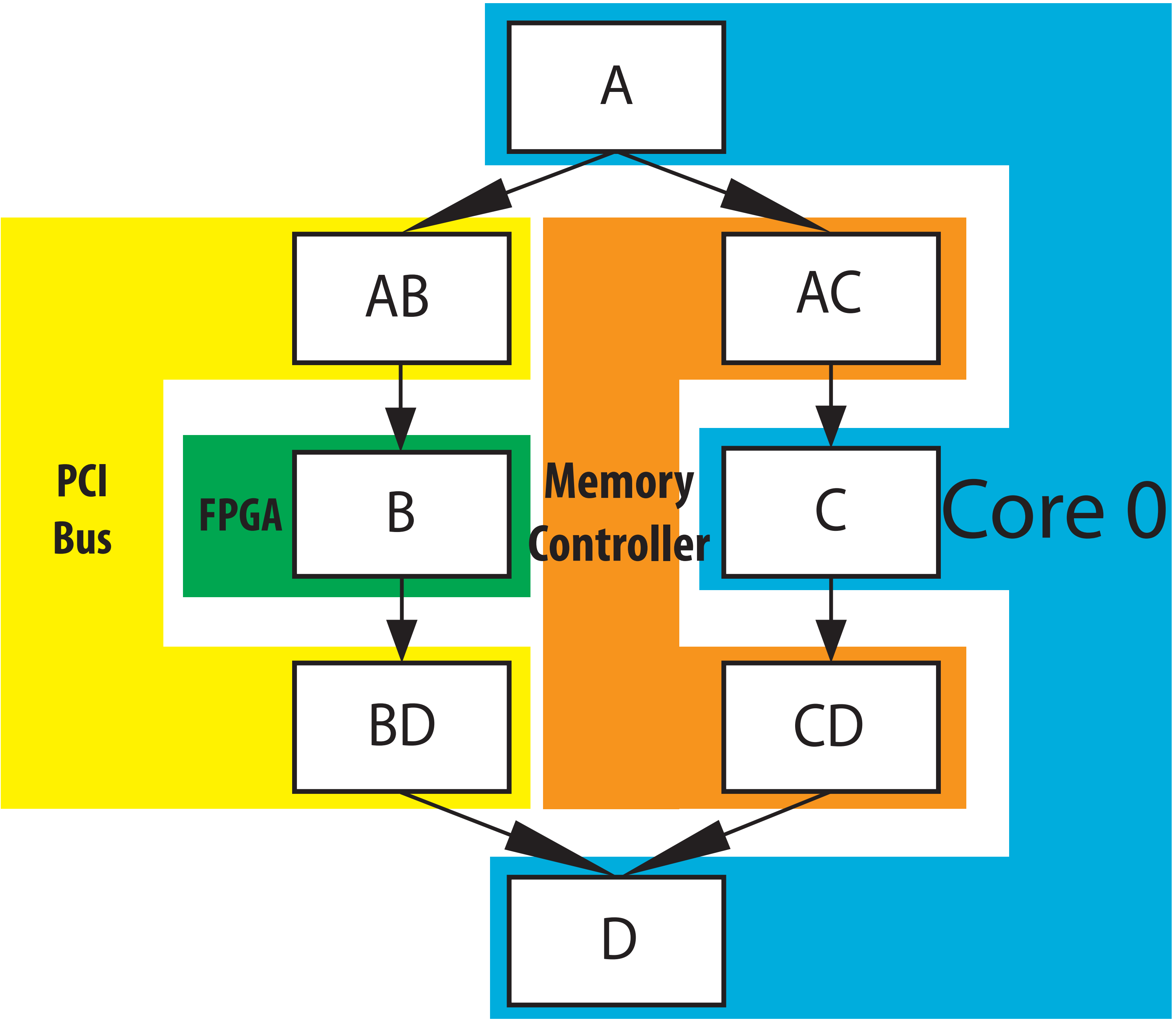
SBS
Stream Based Supercomputing Lab
http://sbs.wustl.edu

Washington University in St.Louis

Application and Implementations



A Hardware Mapping



SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St.Louis

Hypothesis

Can we calculate achievable throughput and place an upper bound for necessary buffering capacity?

Modeling Assumptions

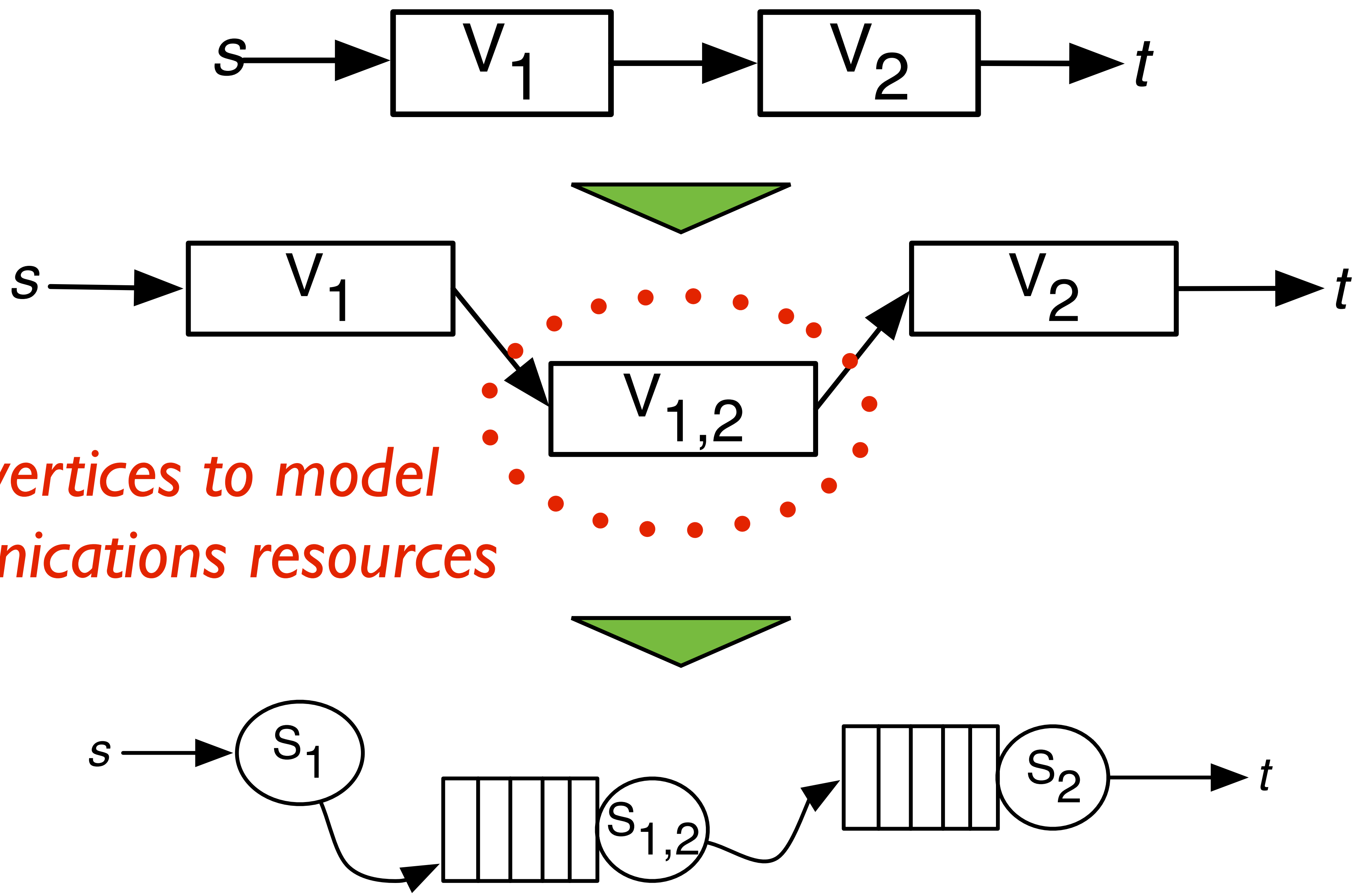
- The system being modeled is at **steady state**
- Arrival process is **Poisson**
- Service times are **exponentially distributed**.
- Buffers are infinite with non-blocking reads and writes.

Overall Model Layout

Application Topology

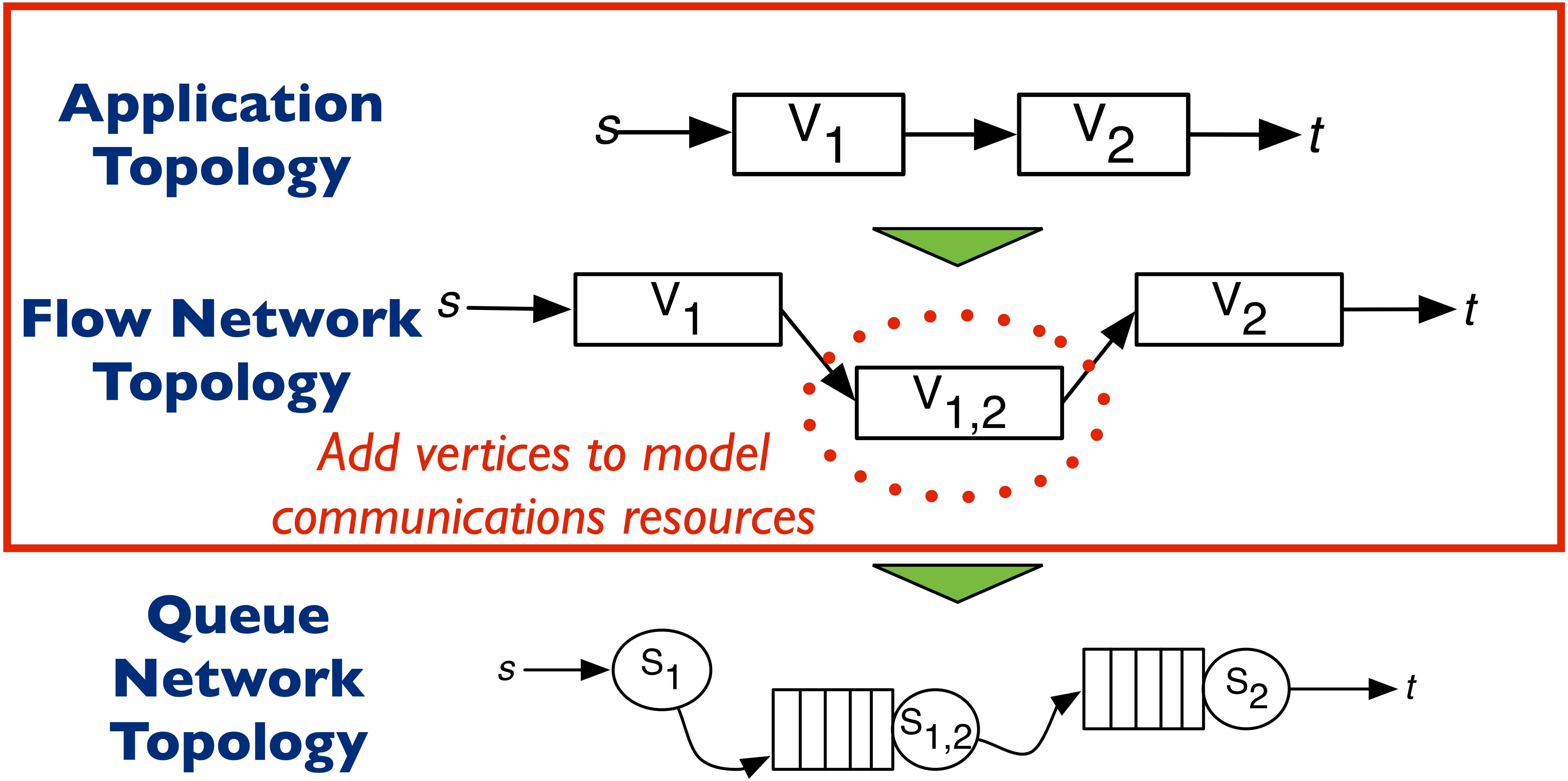
Flow Network Topology

Queue Network Topology



Add vertices to model communications resources

Overall Model Layout



Application Topology

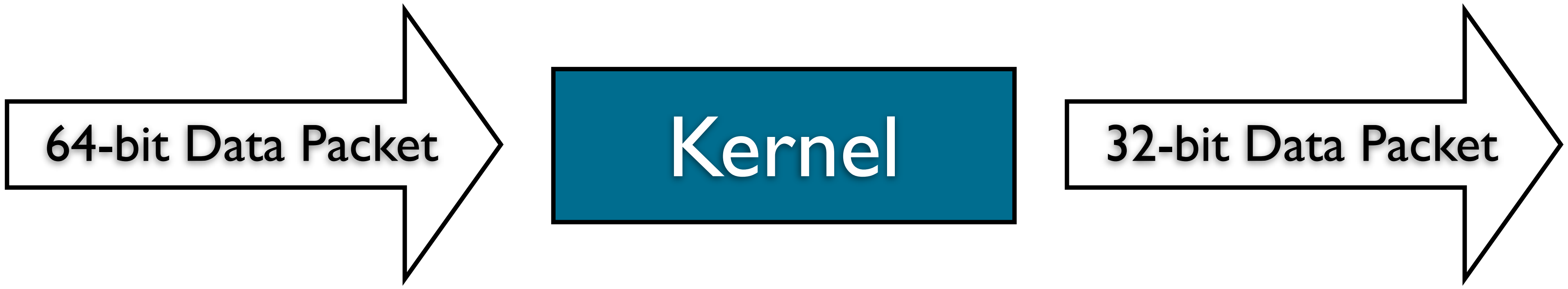
Flow Network Topology

Add vertices to model communications resources

Queue Network Topology

Flow Model Filtering

Filtering - **Gain** or **Loss** of Data



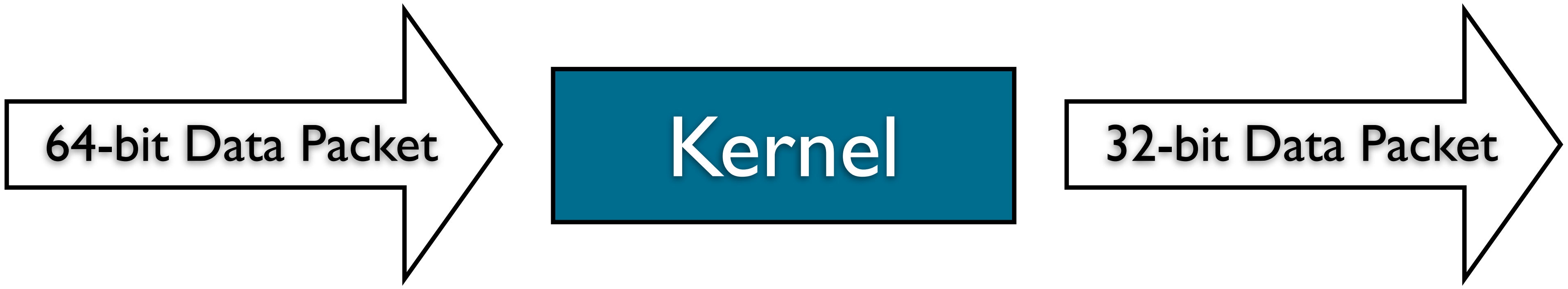
64-bit Data Packet

Kernel

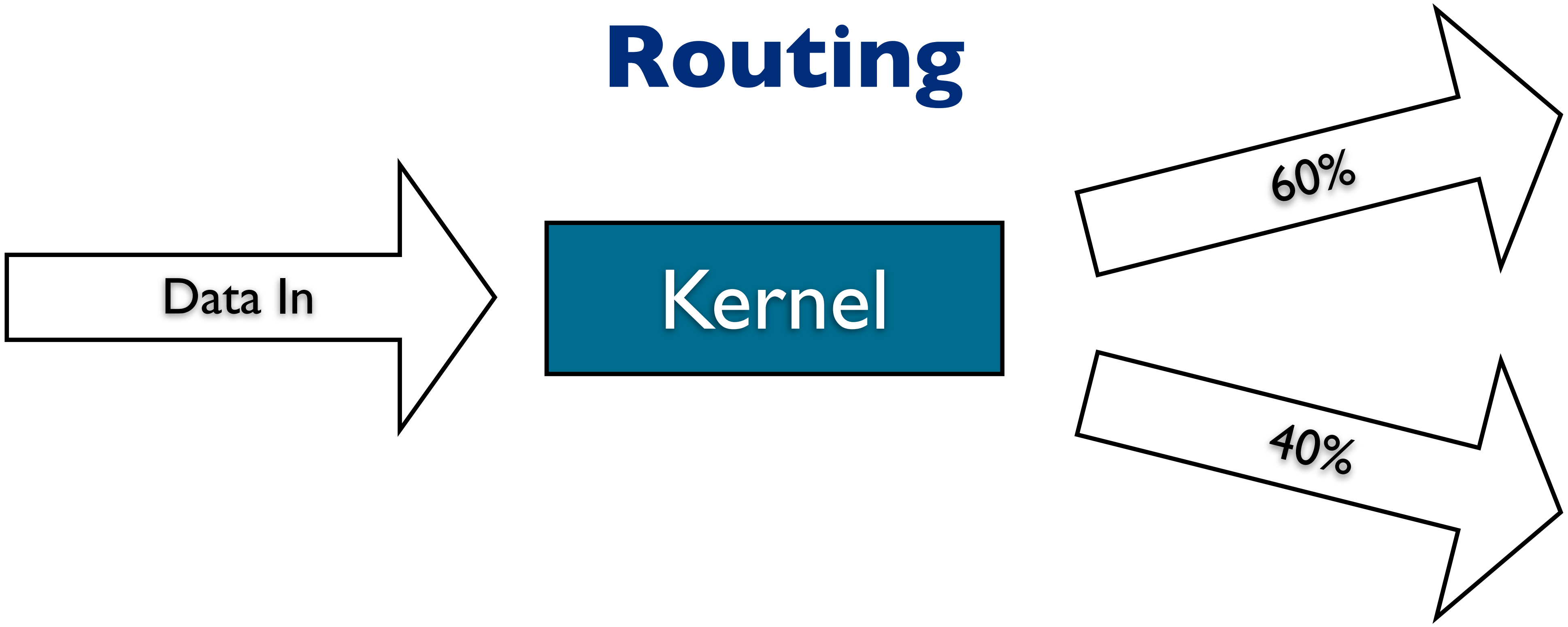
32-bit Data Packet

Flow Model Filtering

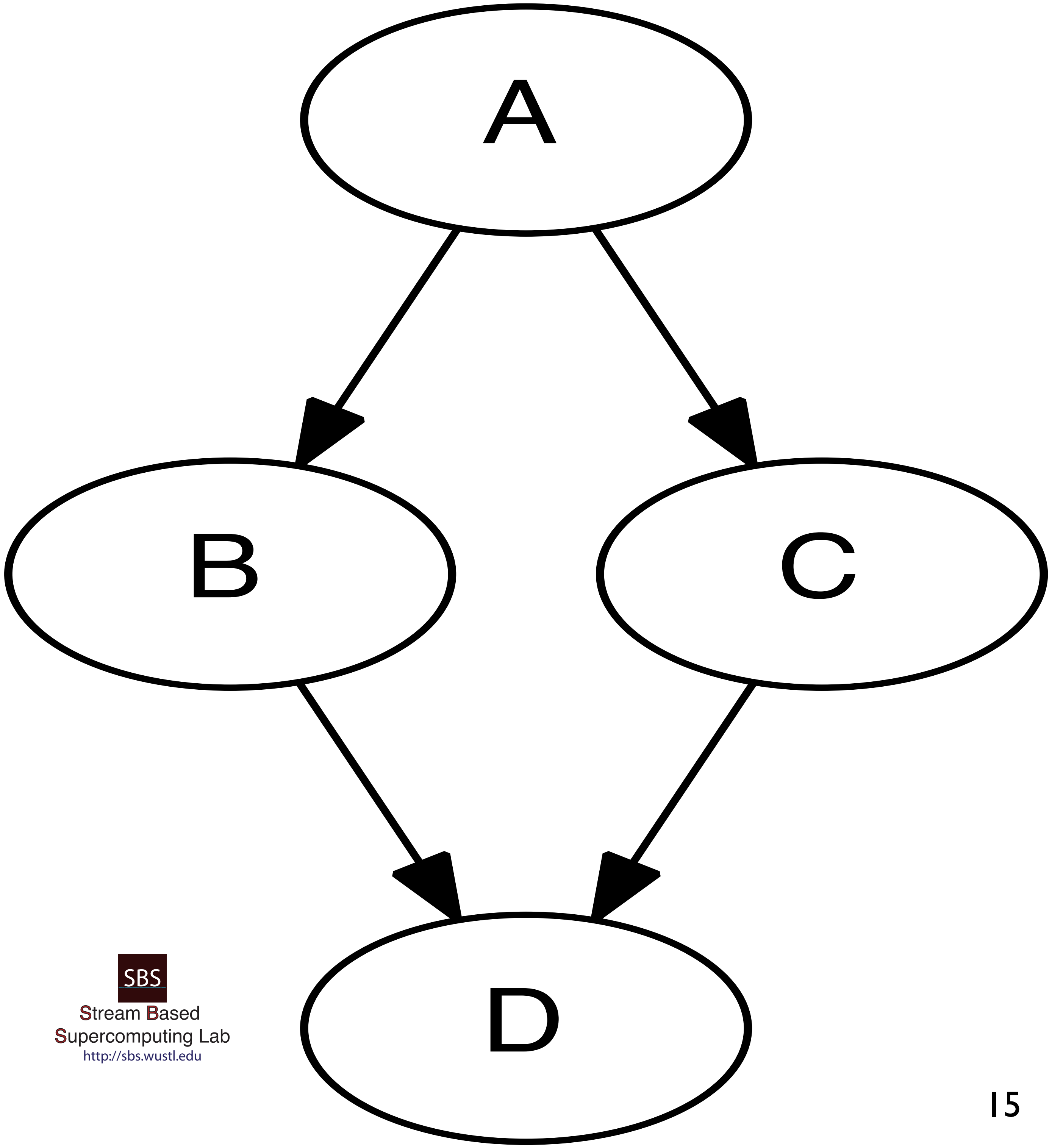
Filtering - Gain or Loss of Data



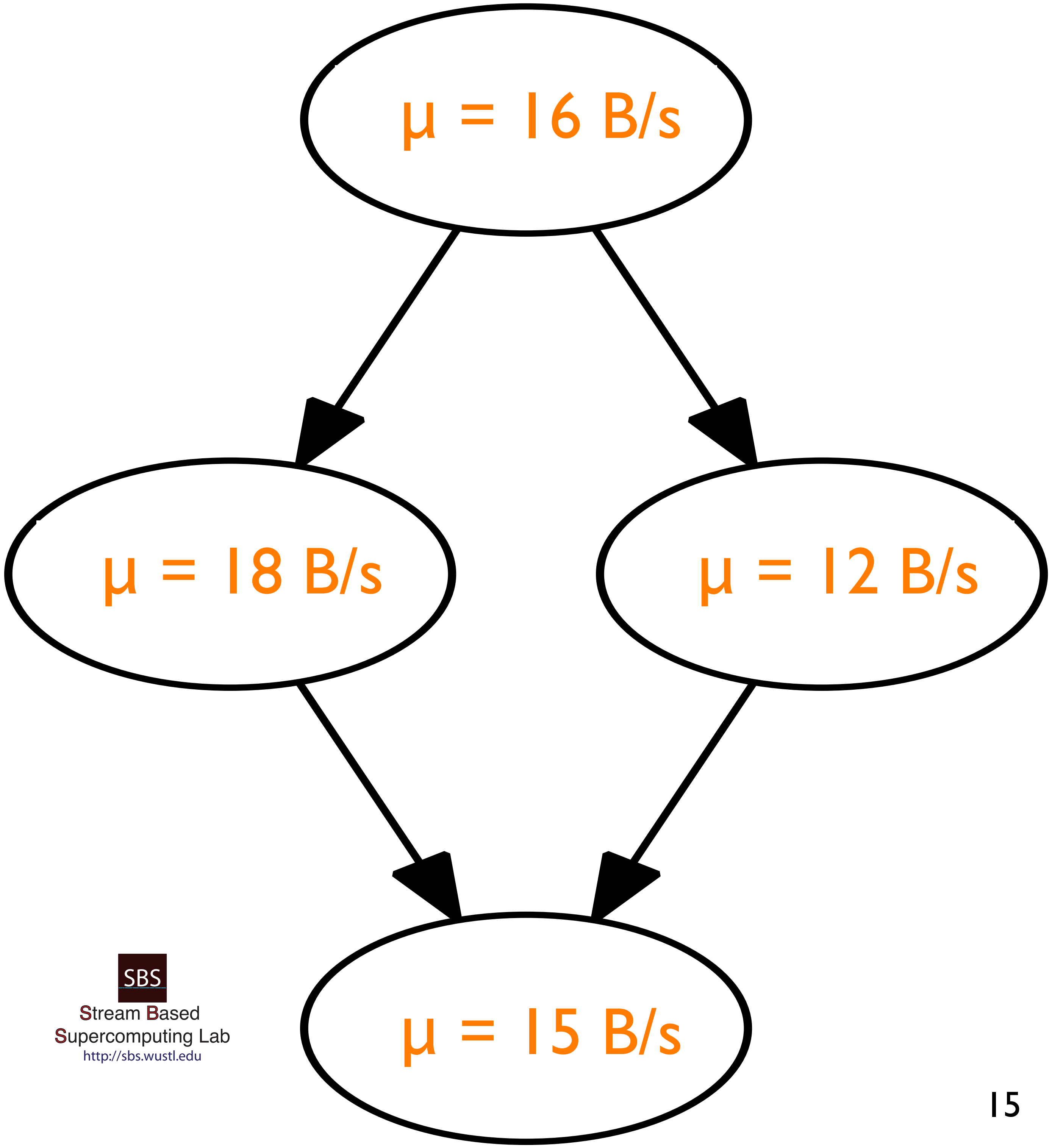
Routing



Flow Model



Flow Model

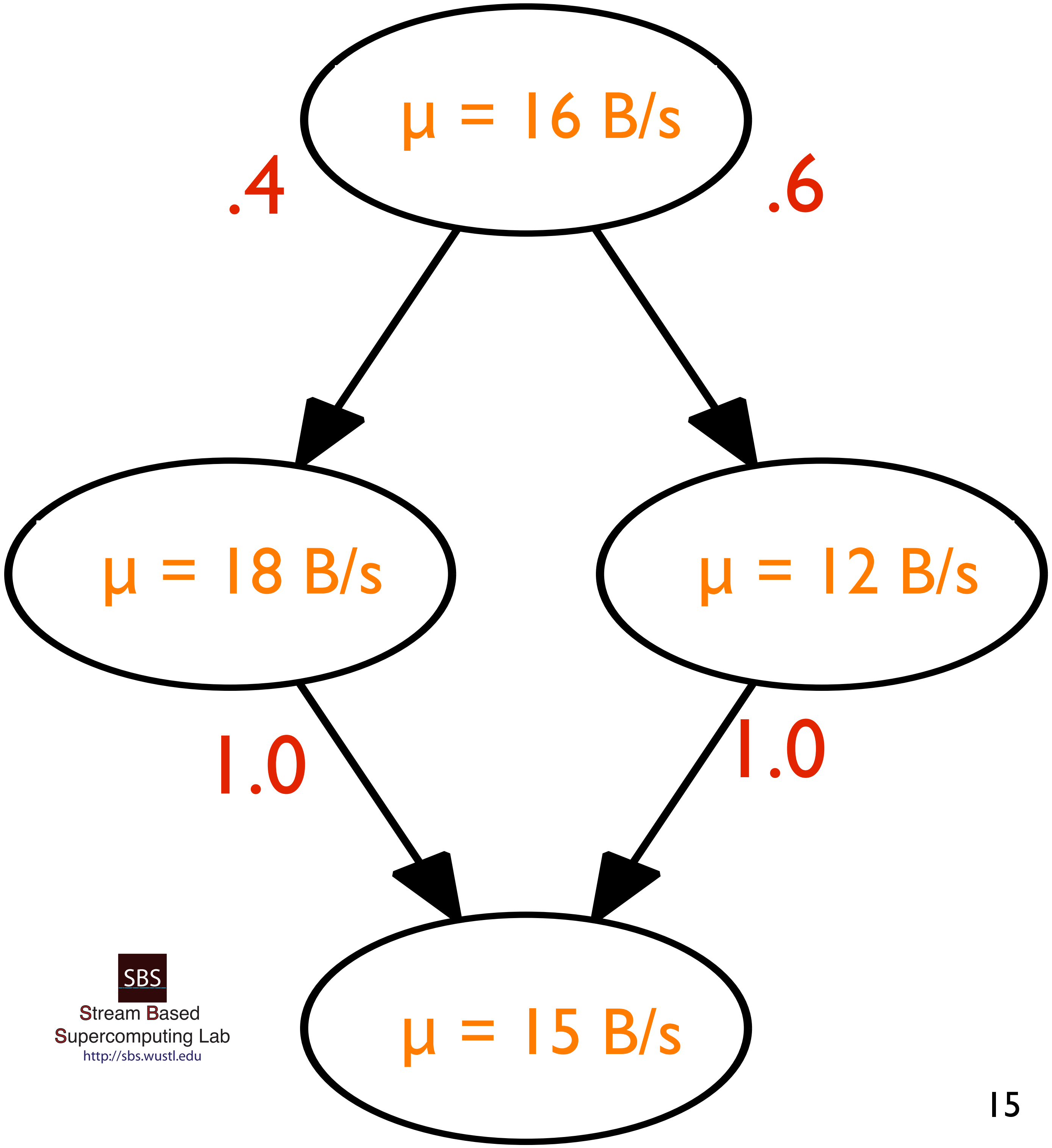


μ - service rate of kernel

SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St. Louis

Flow Model



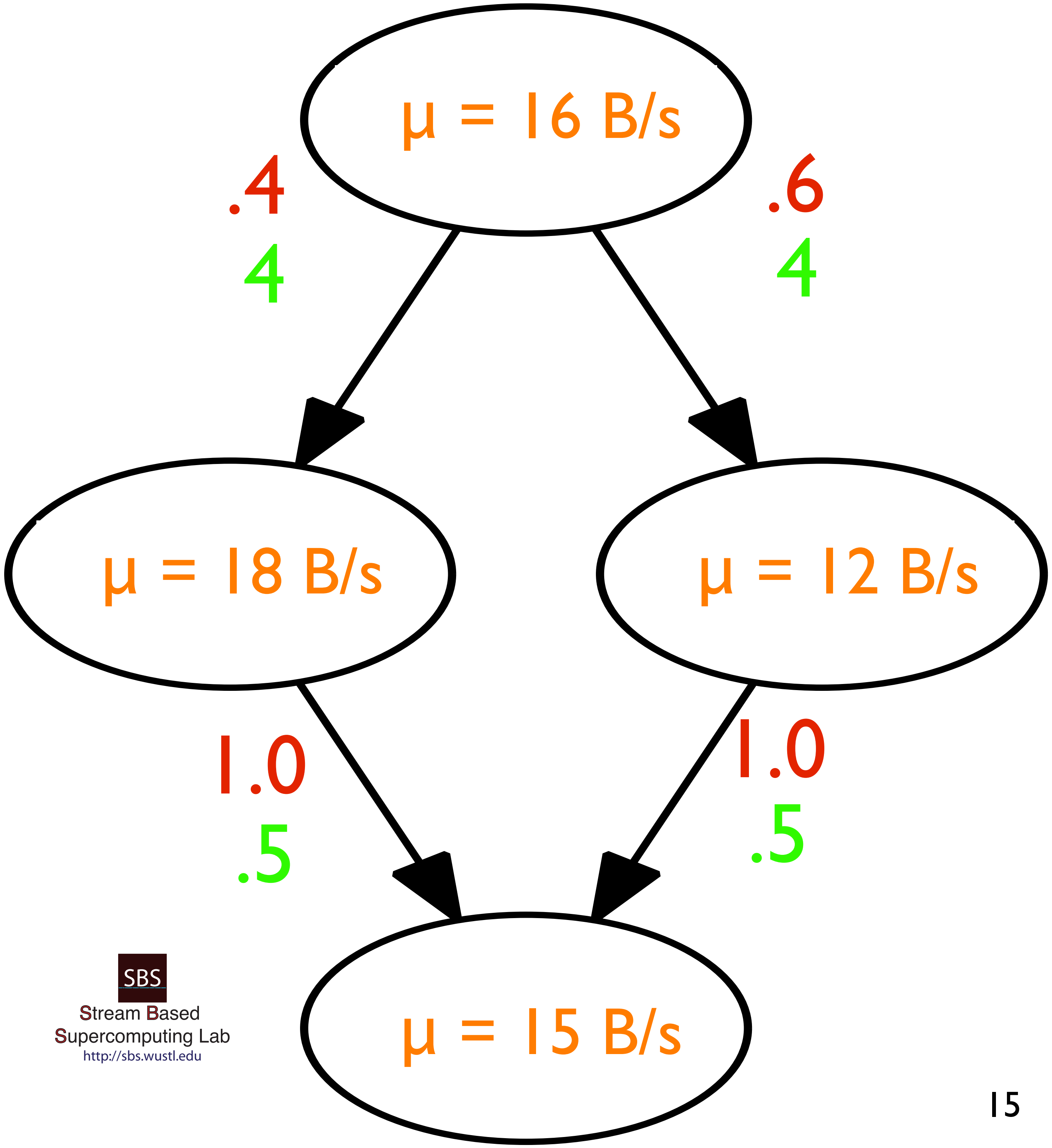
μ - service rate of kernel

F_r- fraction of data along kernel out-edges

SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St.Louis

Flow Model



μ - service rate of kernel

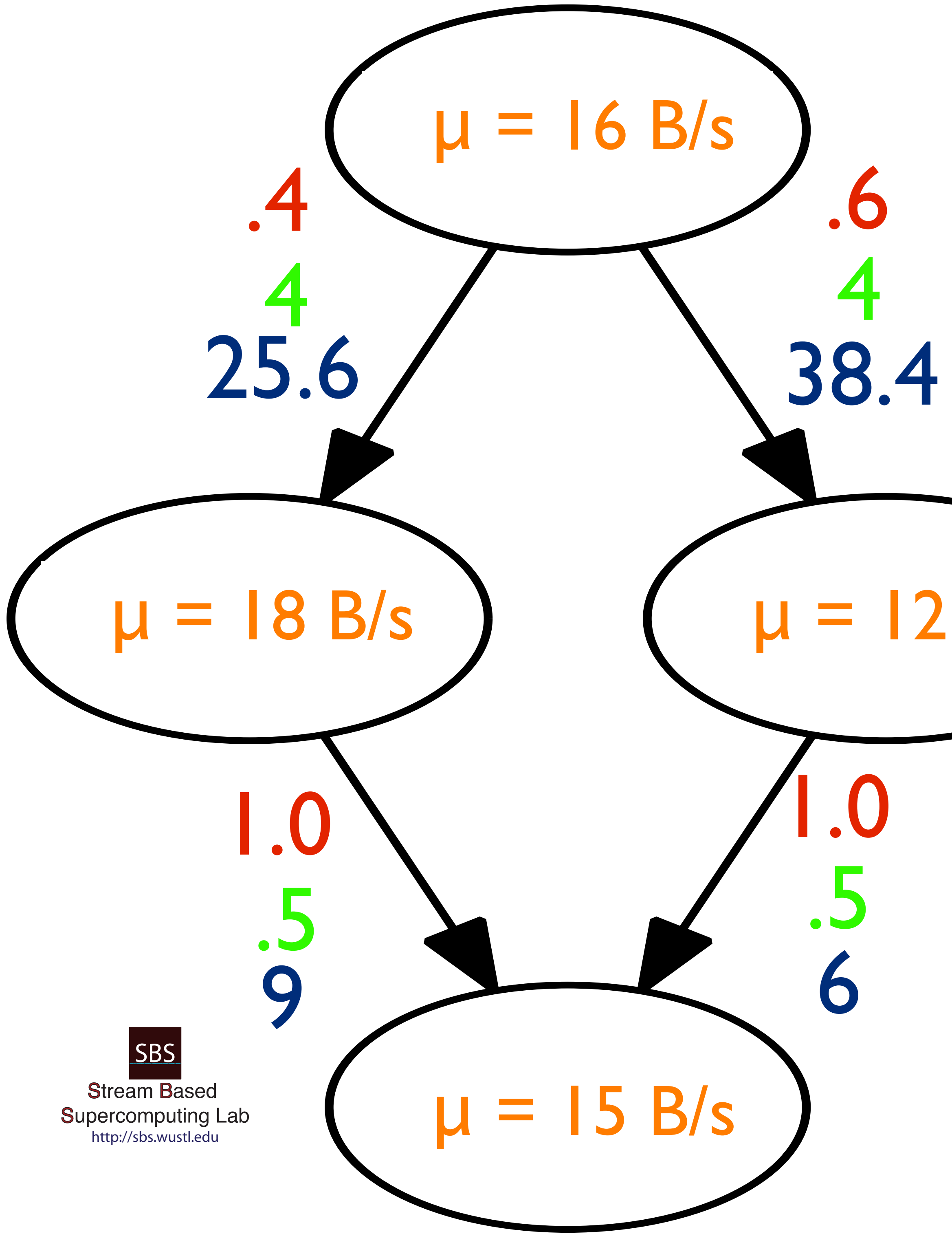
F_r- fraction of data along kernel out-edges

γ - gain function of upstream kernel

SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St.Louis

Flow Model



C - capacity for each edge product of:

μ - service rate of kernel

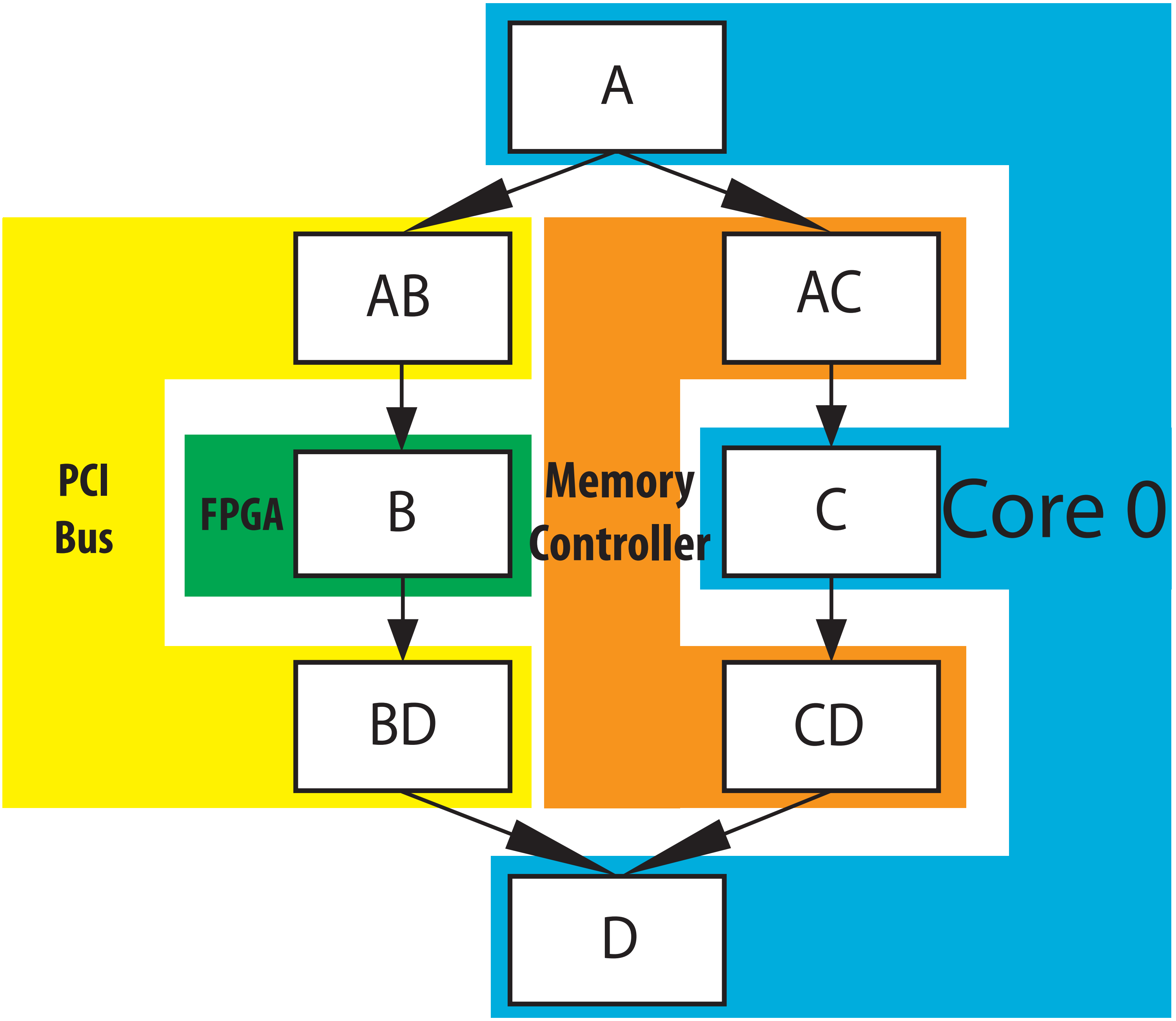
F_r- fraction of data along kernel out-edges

γ - gain function of upstream kernel

SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St. Louis

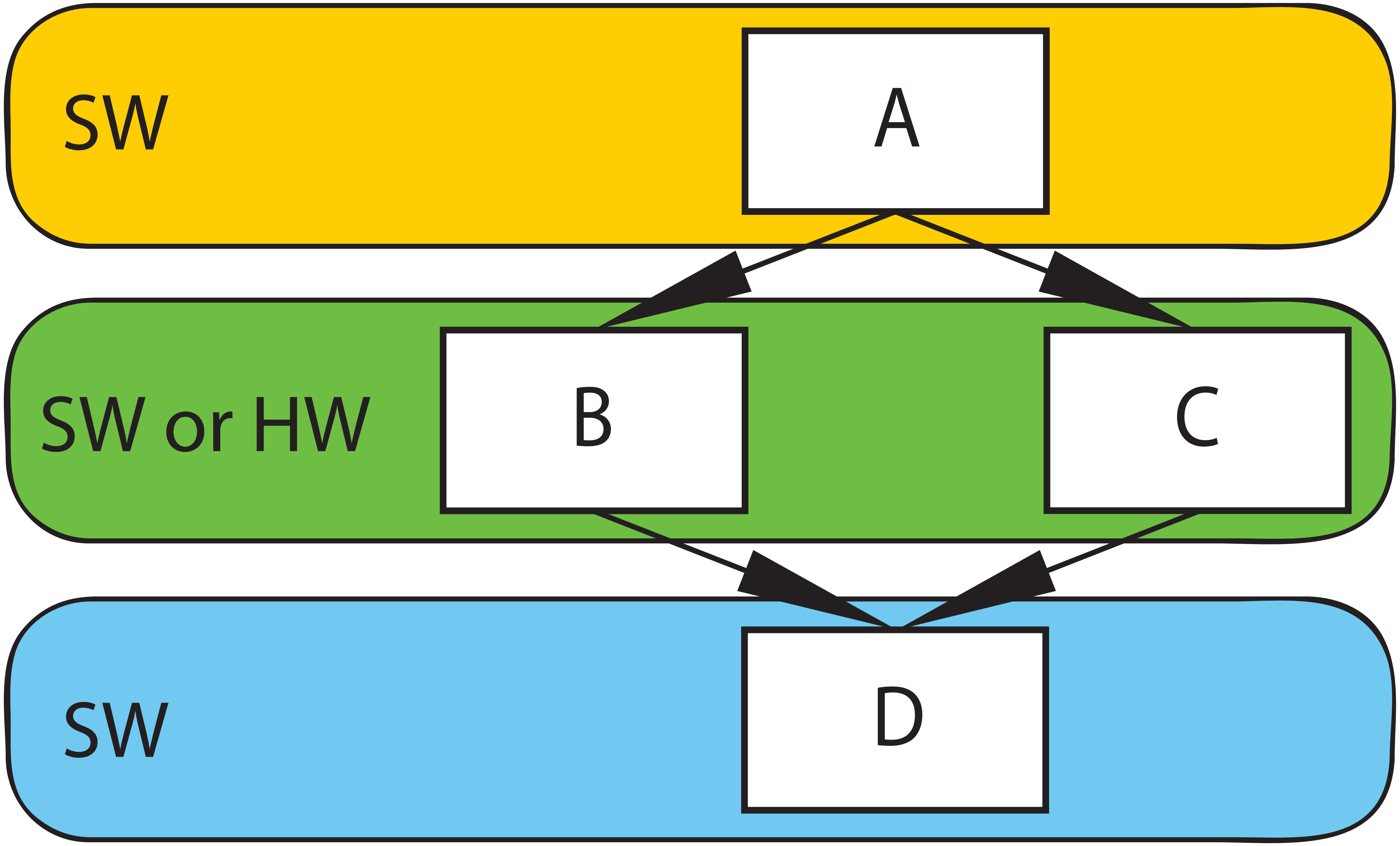
What about sharing?



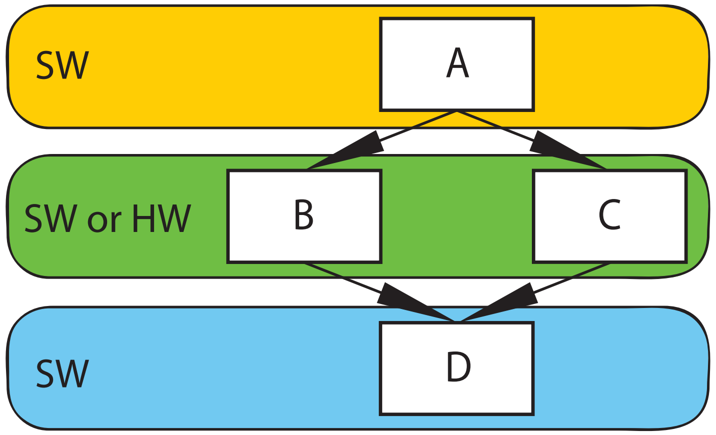
What about sharing?

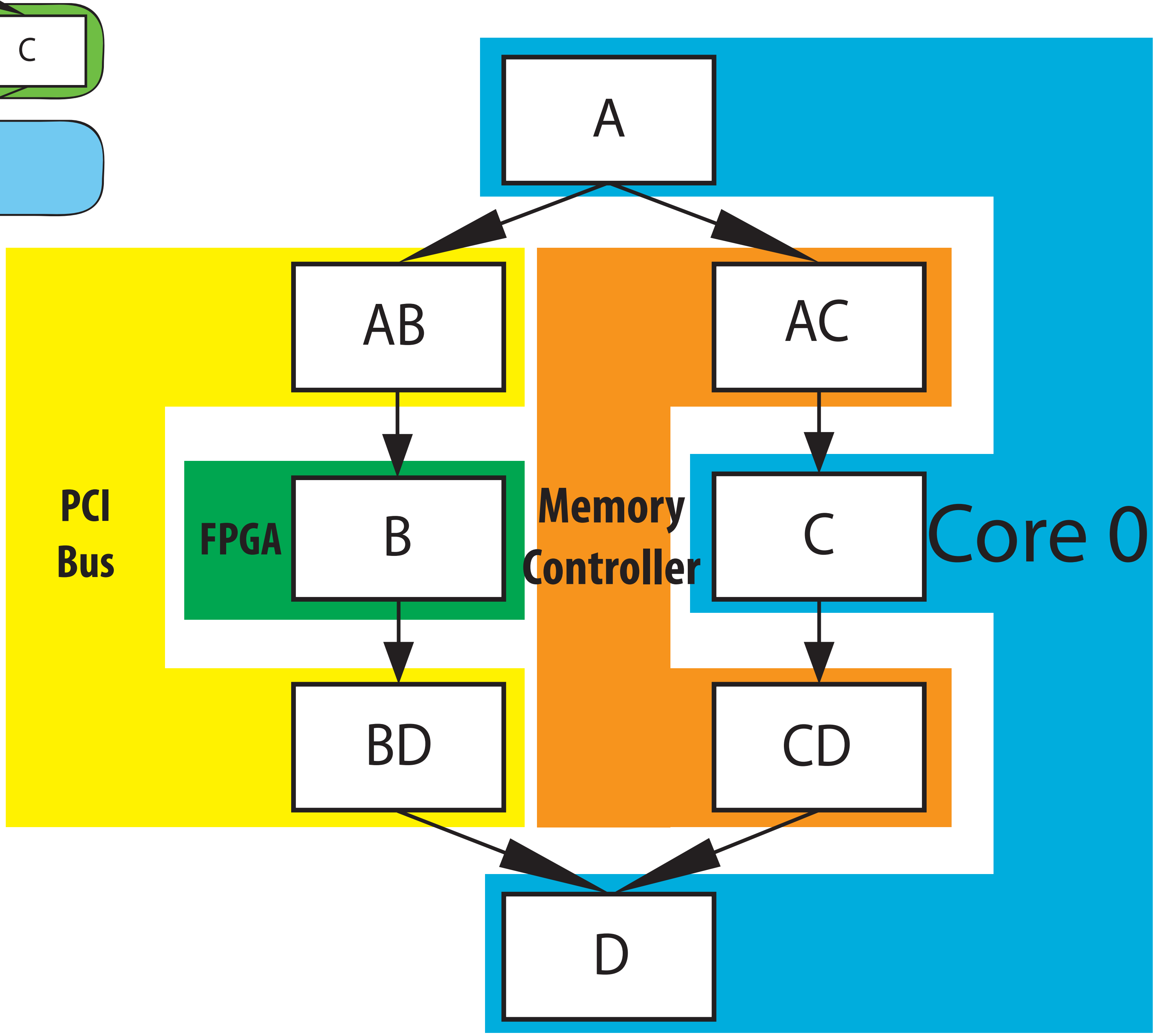
- **Multicore(s)** - Fair Sharing, even division of processing capacity
- **FPGA(s)** - are shared non-temporally via area
- **PCI Bus** - Fair Sharing, even division of bandwidth

Our Example

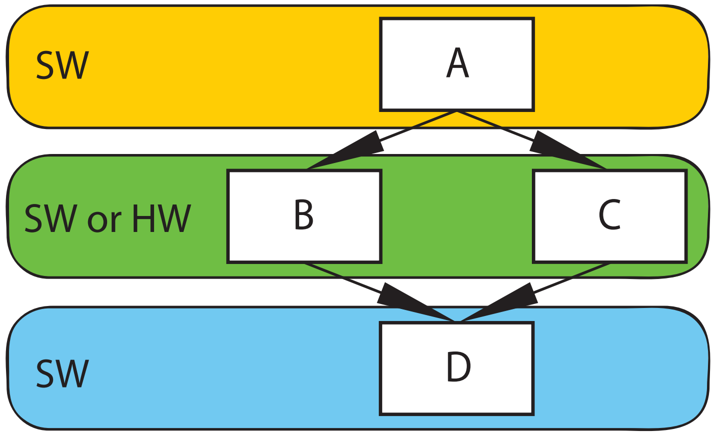


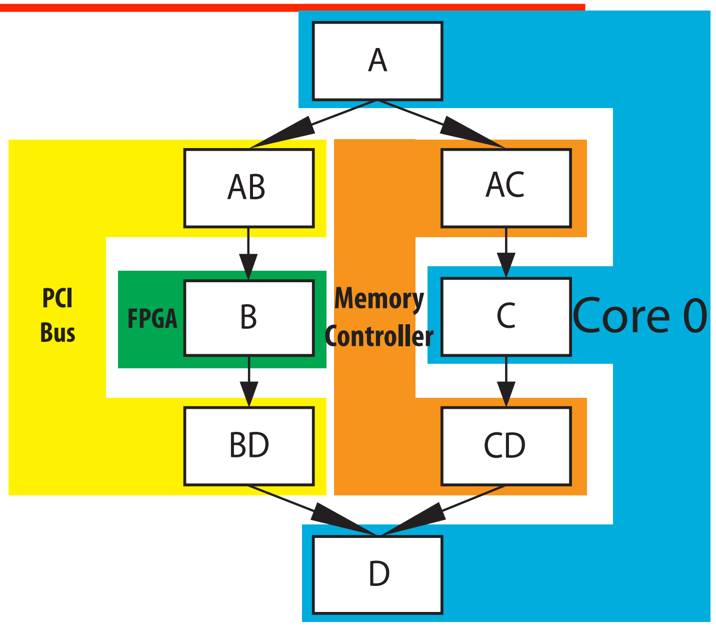
Our Example

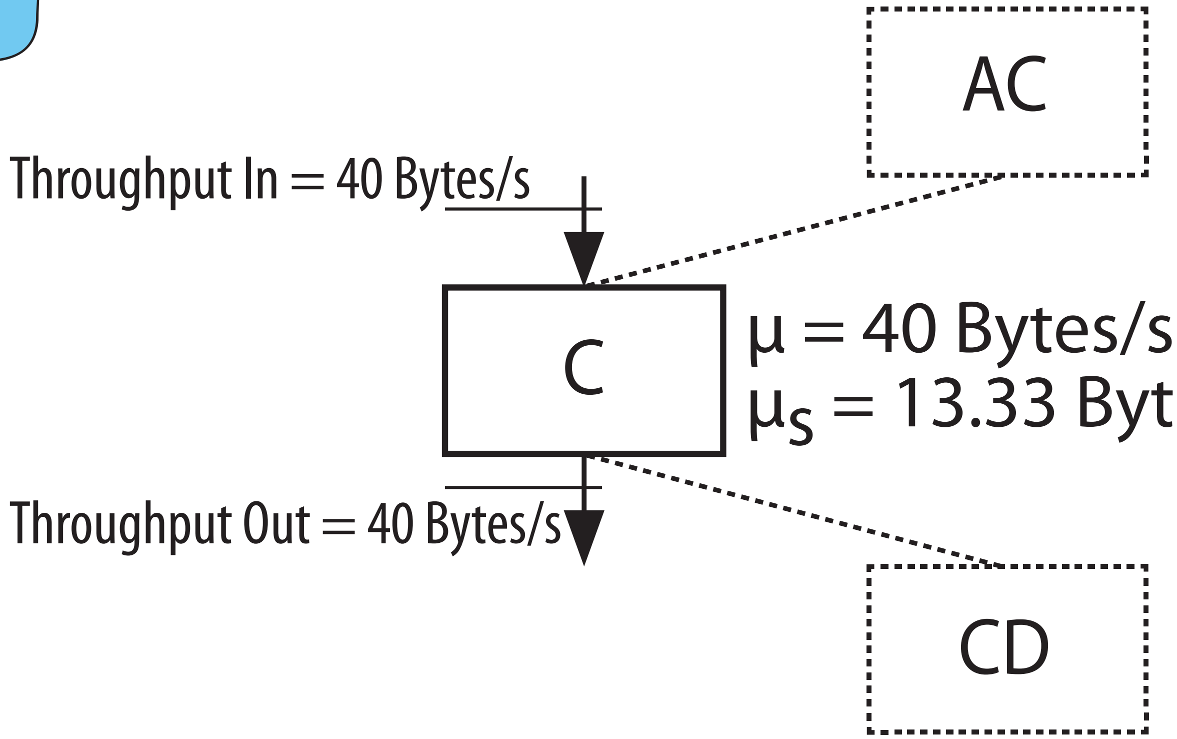

SW
A
SW or HW
B
C
SW
D


A
AB
AC
PCI
Bus
FPGA
B
Memory
Controller
C
Core 0
BD
CD
D

Our Example







$\mu = 40$ Bytes/s

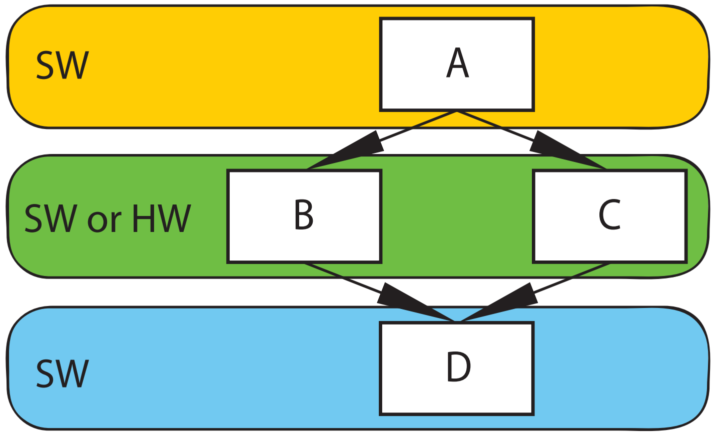
$\mu_S = 13.33$ Bytes/s

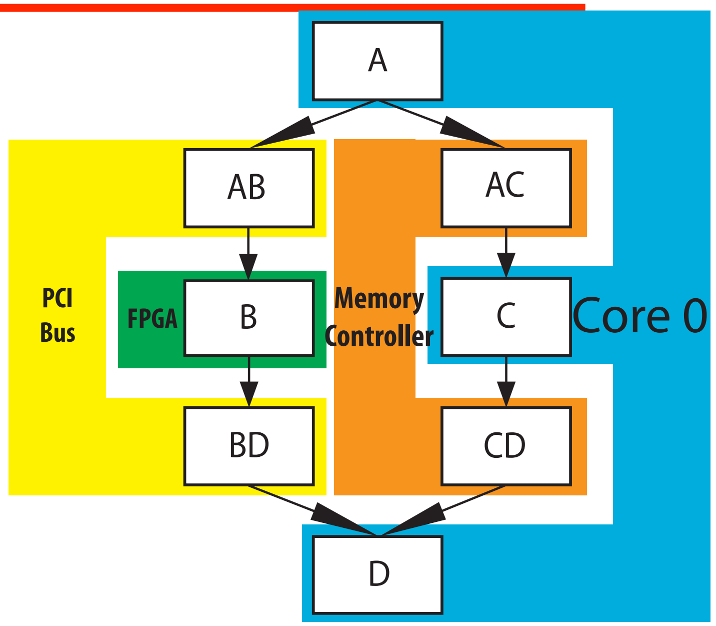
Routing Fraction (F_r)=1.0

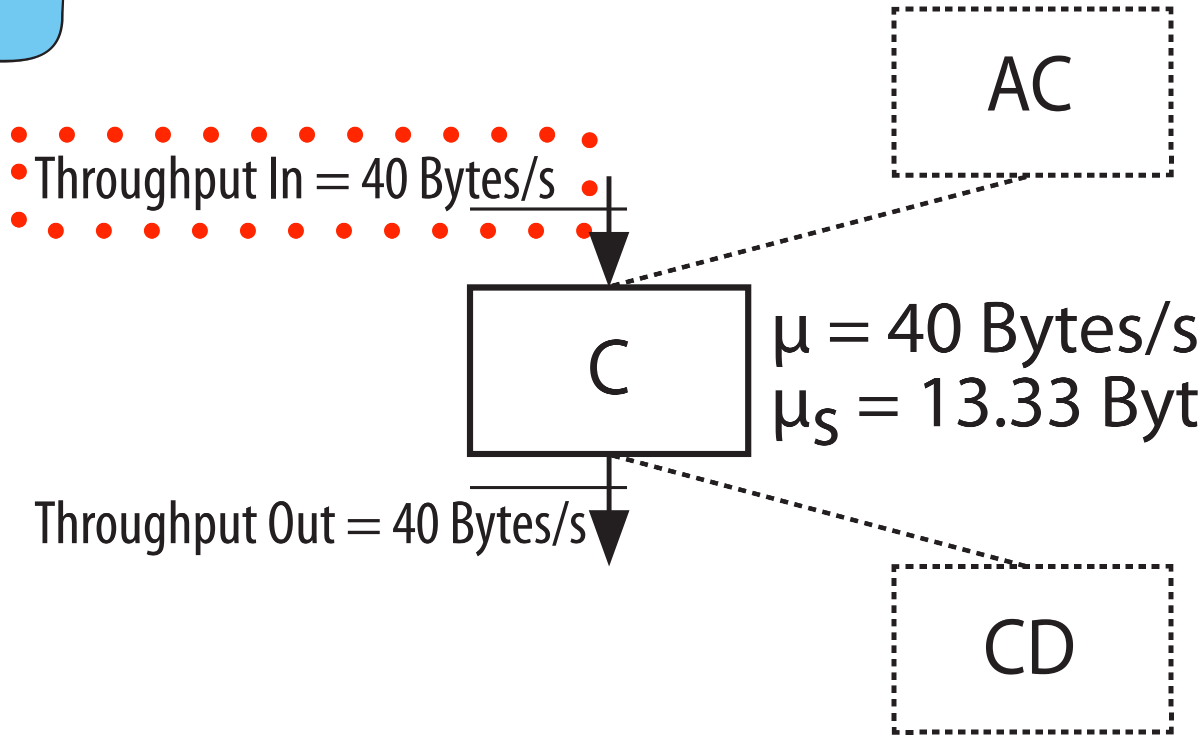
Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Our Example







Throughput In = 40 Bytes/s

μ = 40 Bytes/s

μ_s = 13.33 Bytes/s

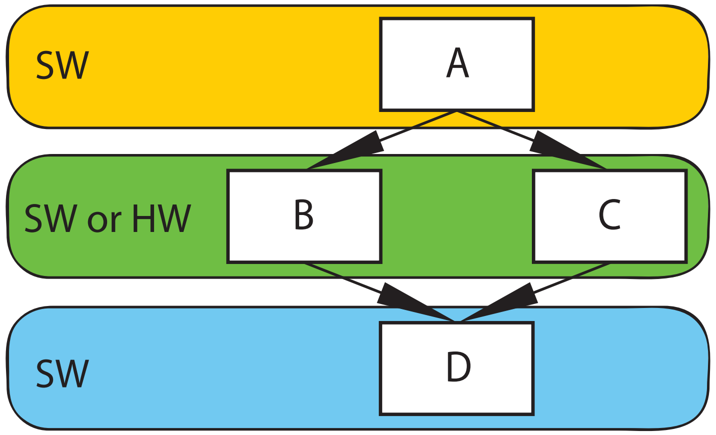
Throughput Out = 40 Bytes/s

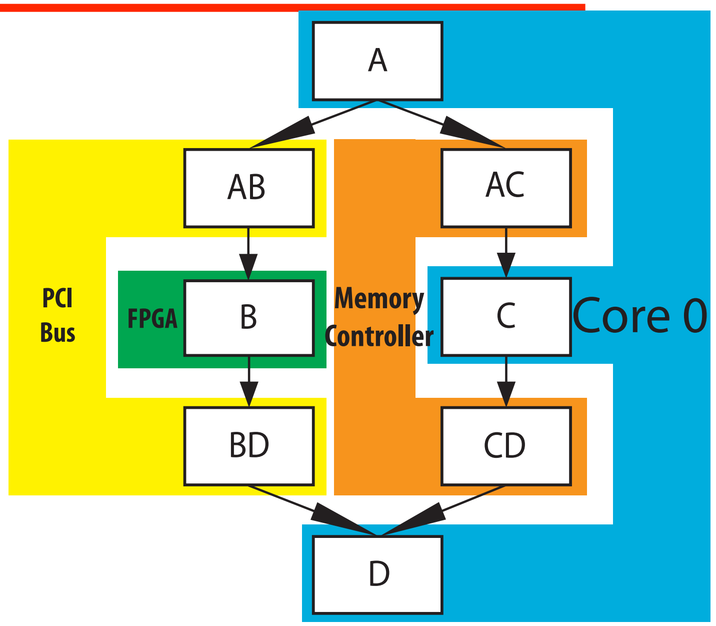
Routing Fraction (F_r)=1.0

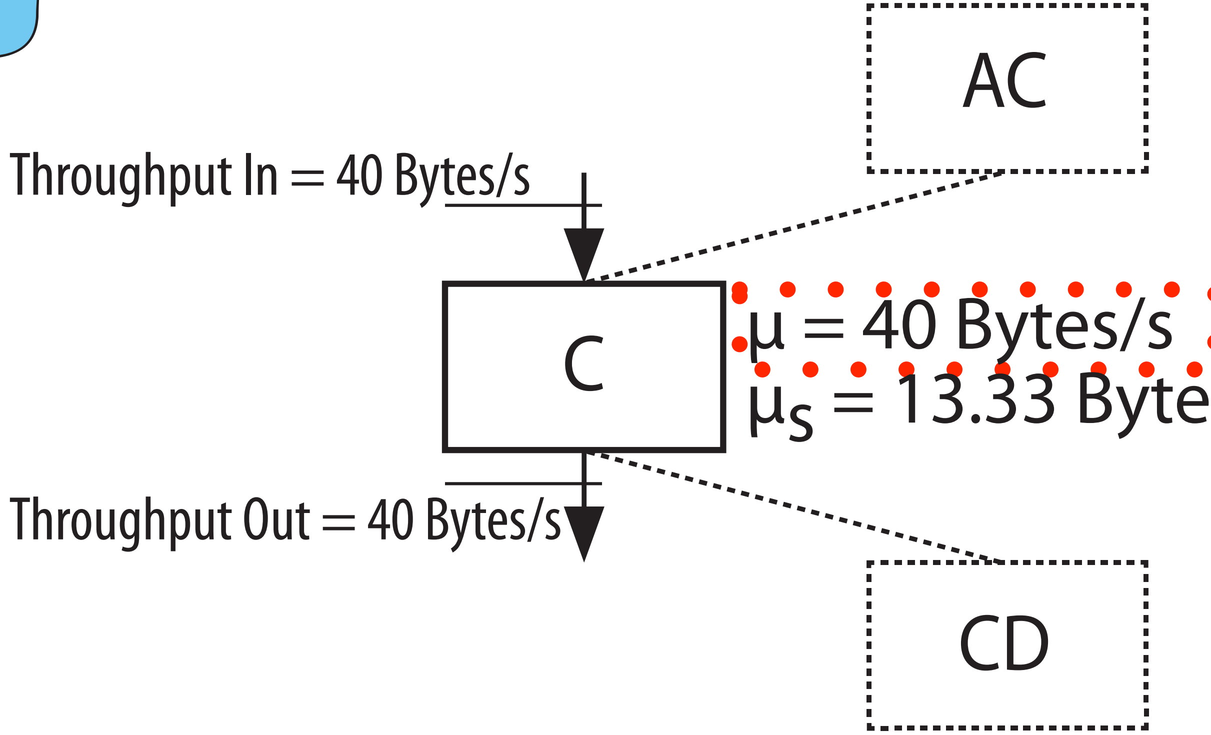
Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Our Example







$\mu = 40$ Bytes/s

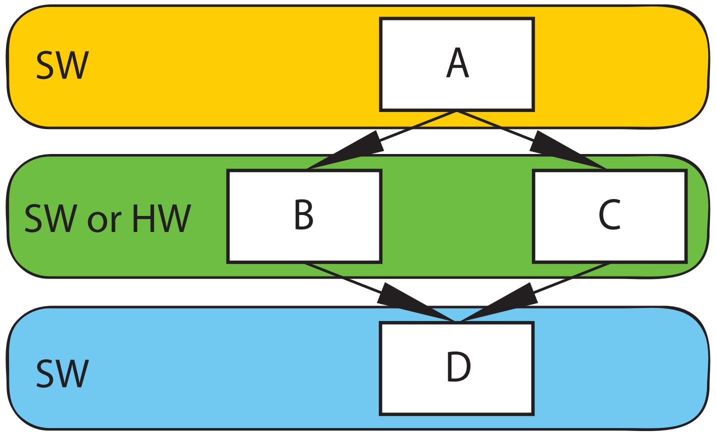
$\mu_s = 13.33$ Bytes/s

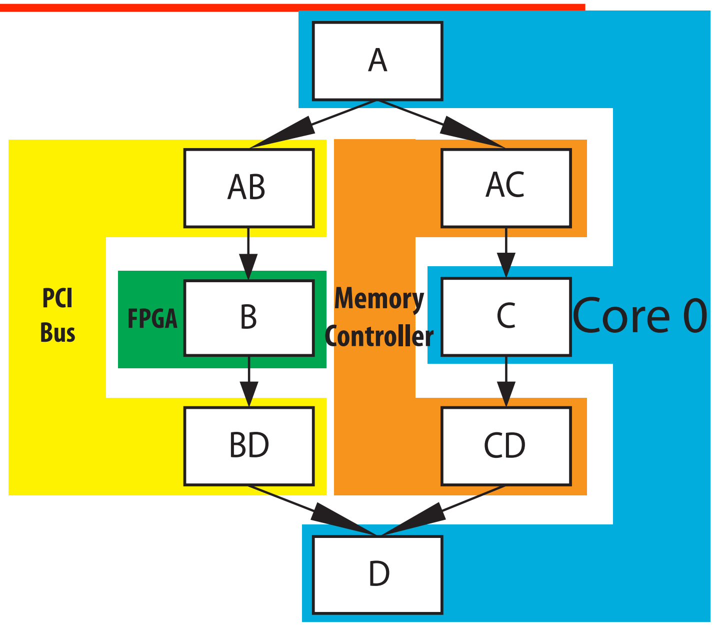
Routing Fraction (F_r)=1.0

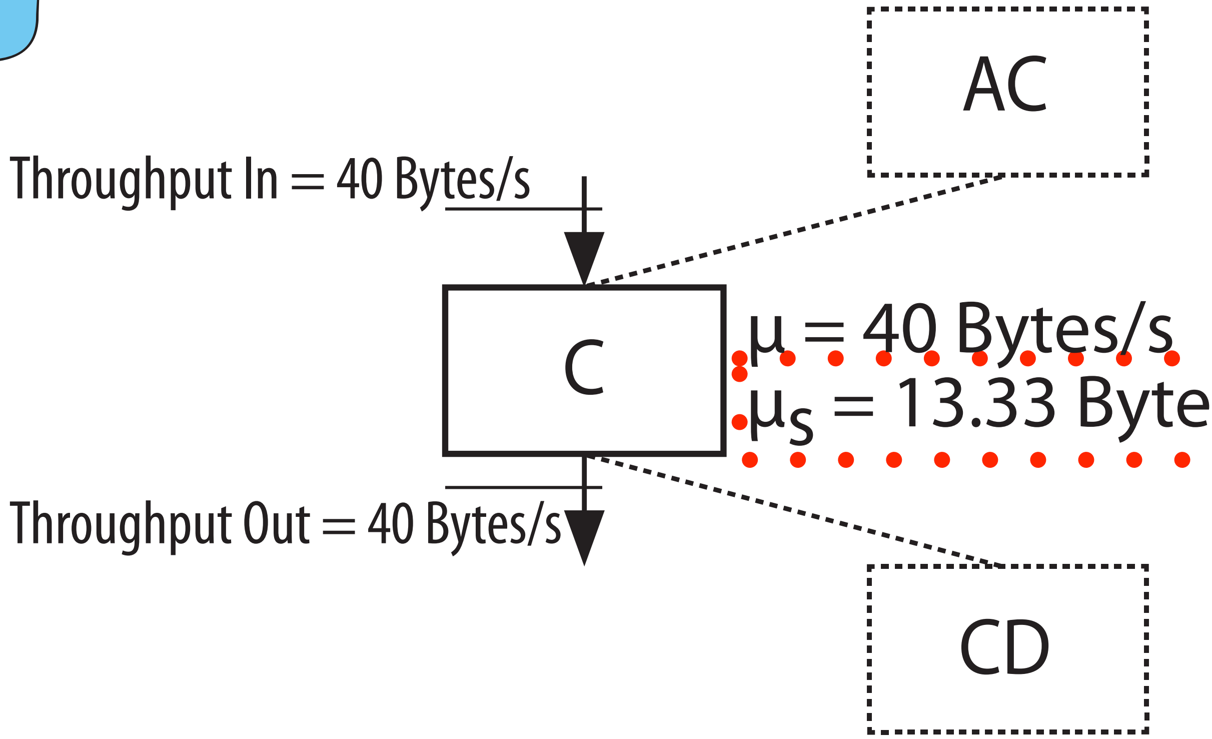
Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Our Example







Throughput In = 40 Bytes/s

μ = 40 Bytes/s

μ_s = 13.33 Bytes/s

Throughput Out = 40 Bytes/s

Routing Fraction (F_r)=1.0

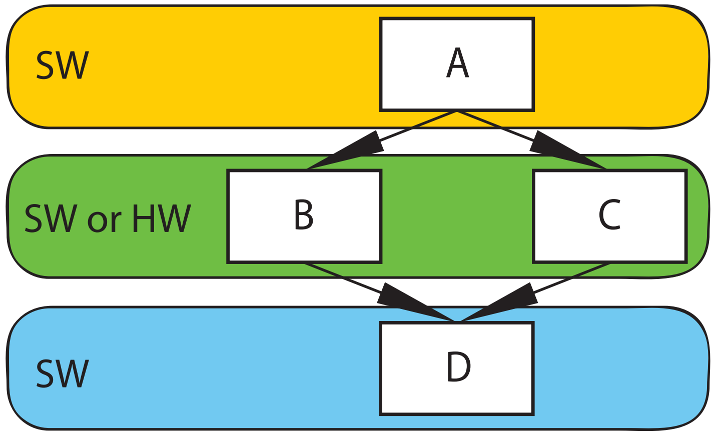
Gain Function (γ)=1.0

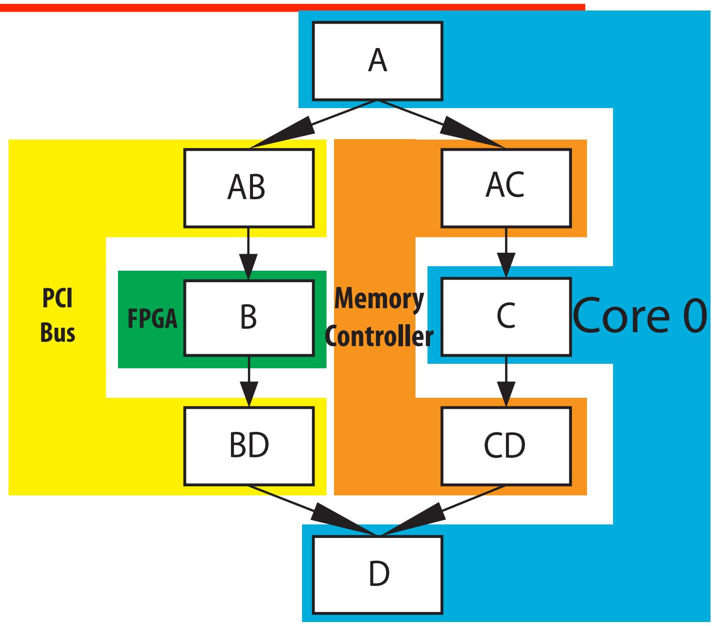
Expected Departure Rate (E_D)=13.33 Bytes/s

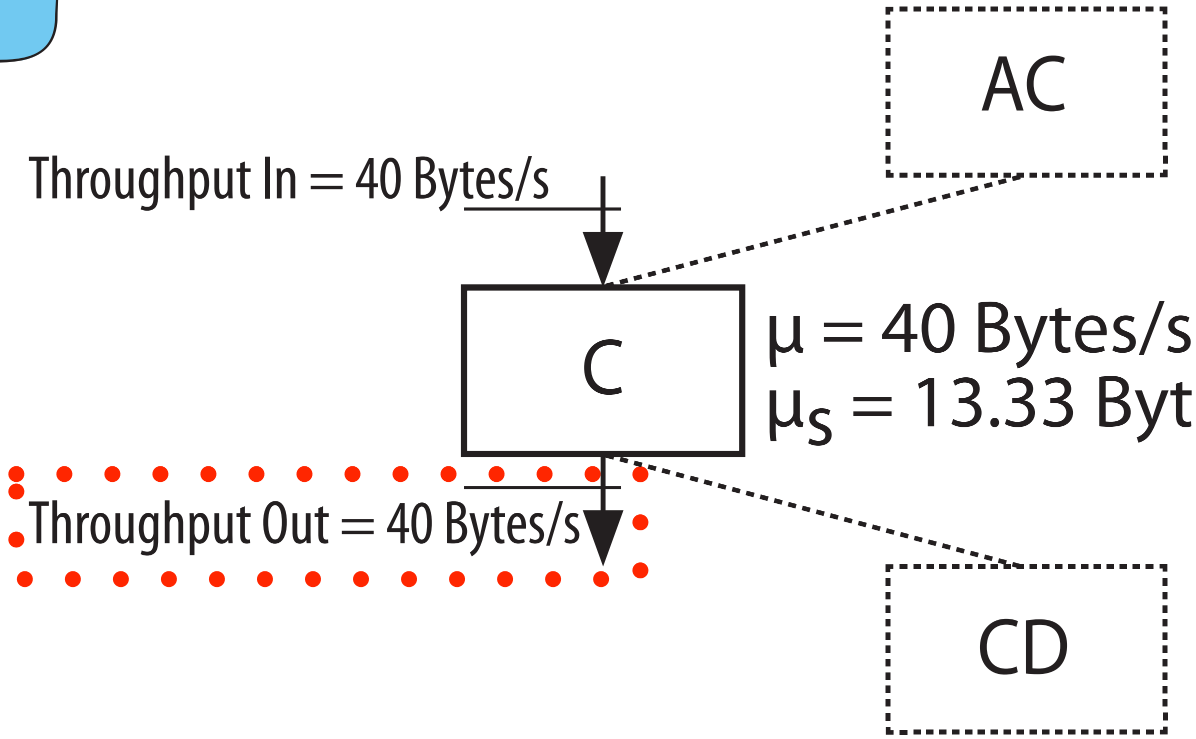
SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St. Louis

Our Example







$\mu = 40$ Bytes/s

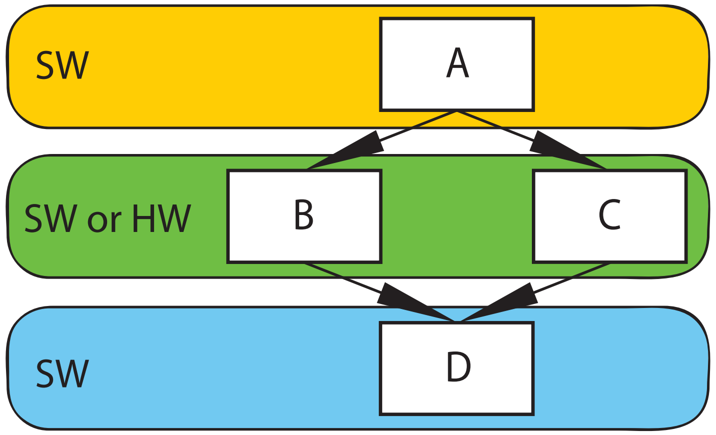
$\mu_S = 13.33$ Bytes/s

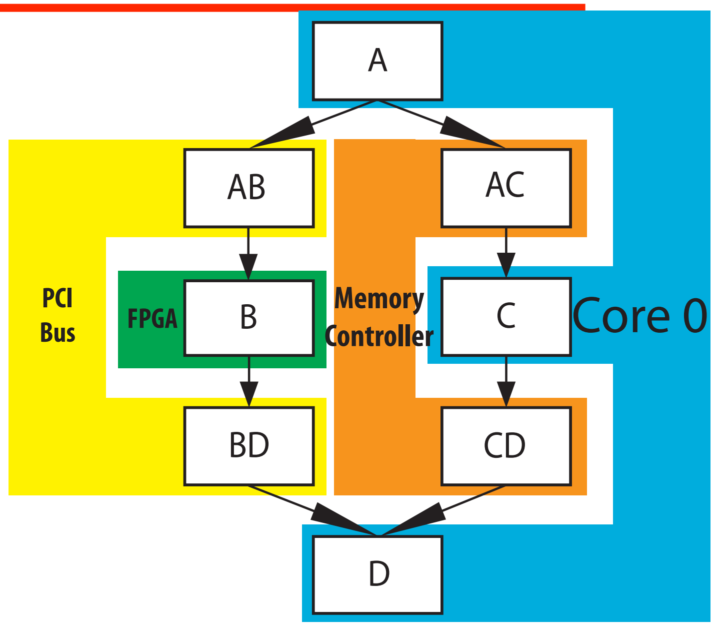
Routing Fraction (F_r)=1.0

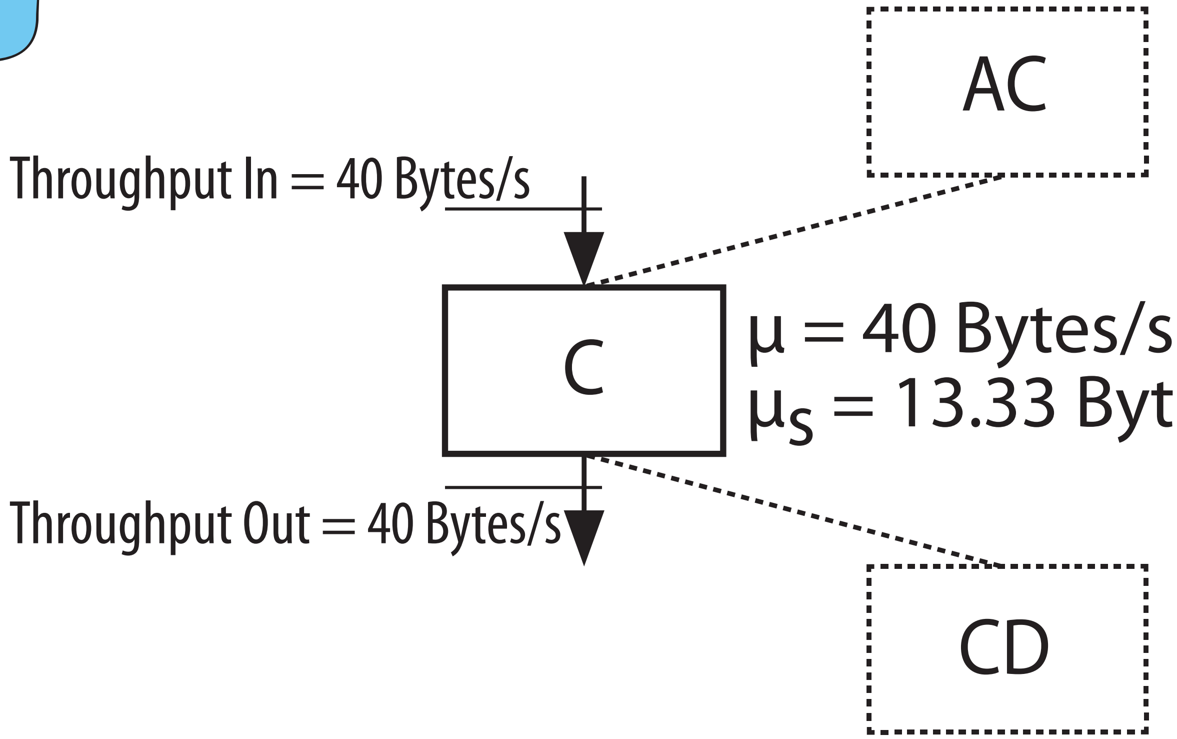
Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Our Example







$\mu = 40$ Bytes/s

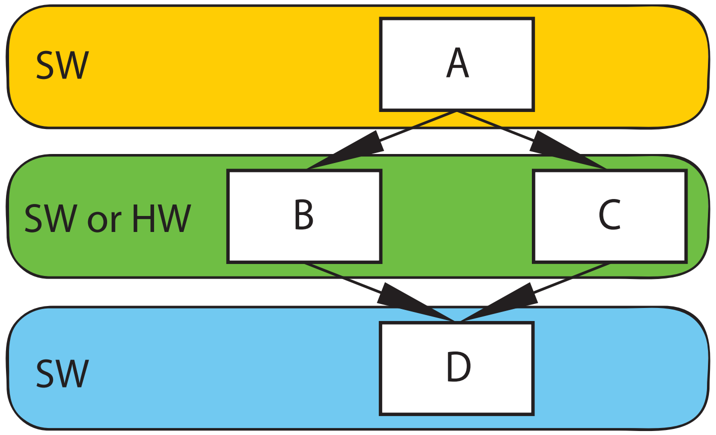
$\mu_S = 13.33$ Bytes/s

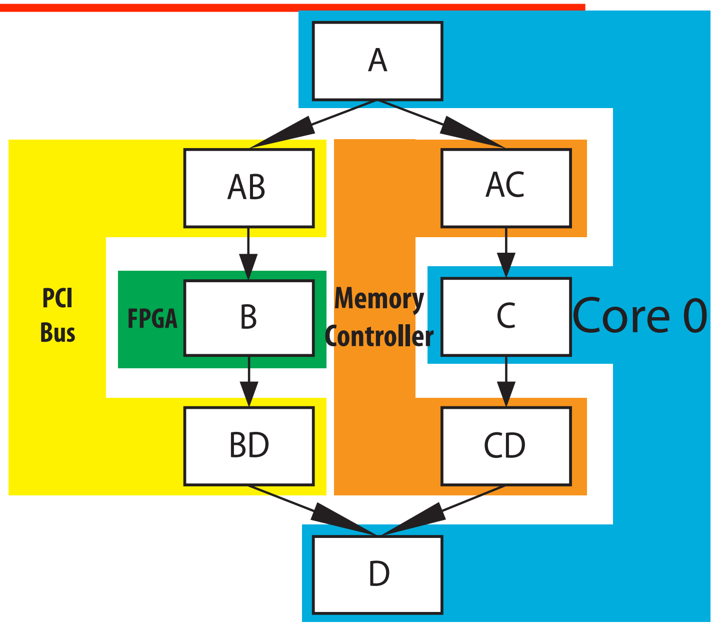
Routing Fraction (F_r)=1.0

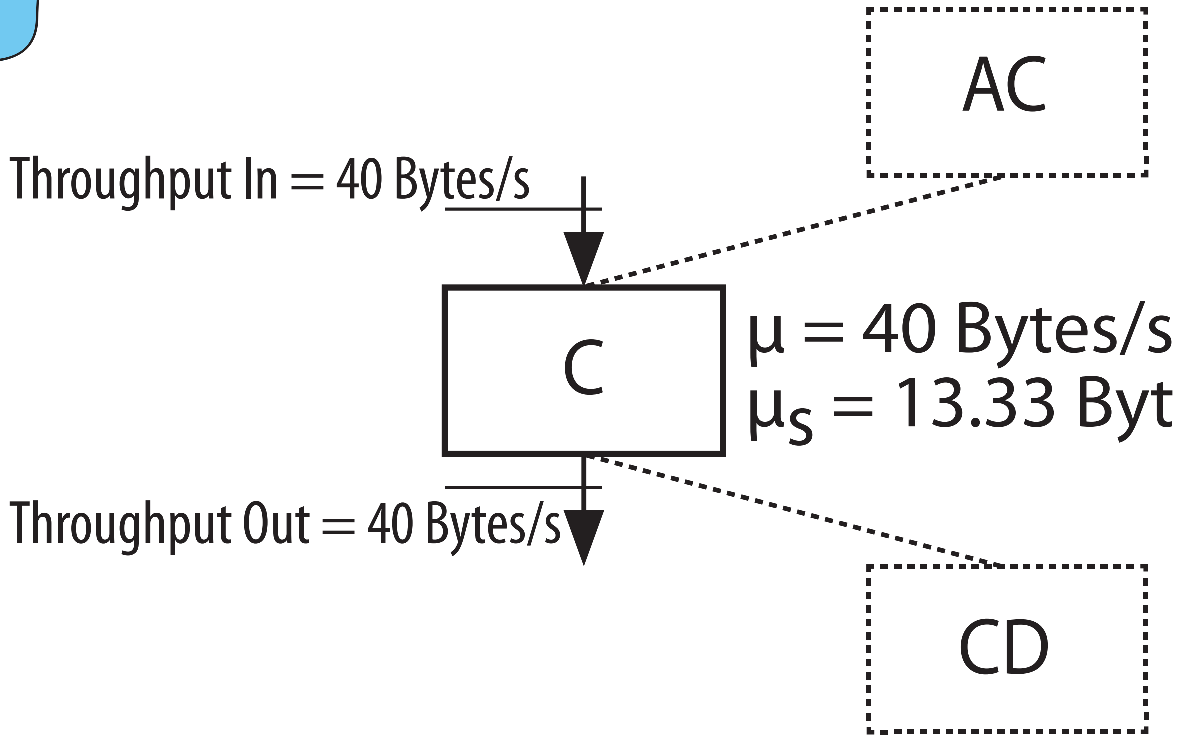
Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Our Example







μ = 40 Bytes/s

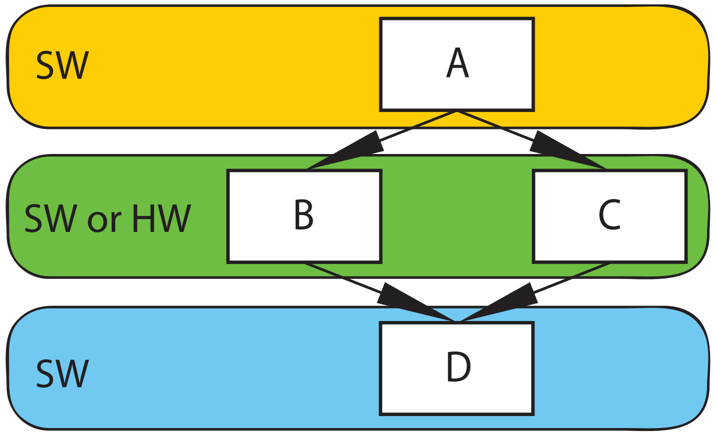
μ_S = 13.33 Bytes/s

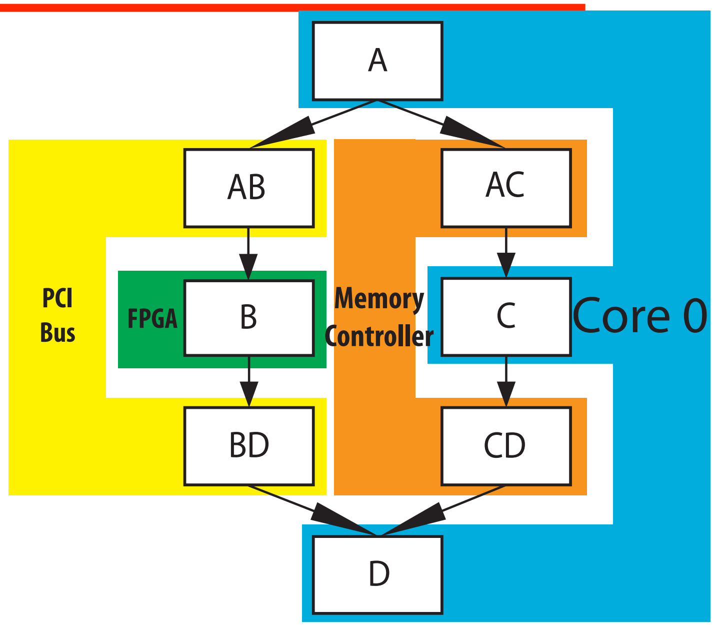
Routing Fraction (F_r)=1.0

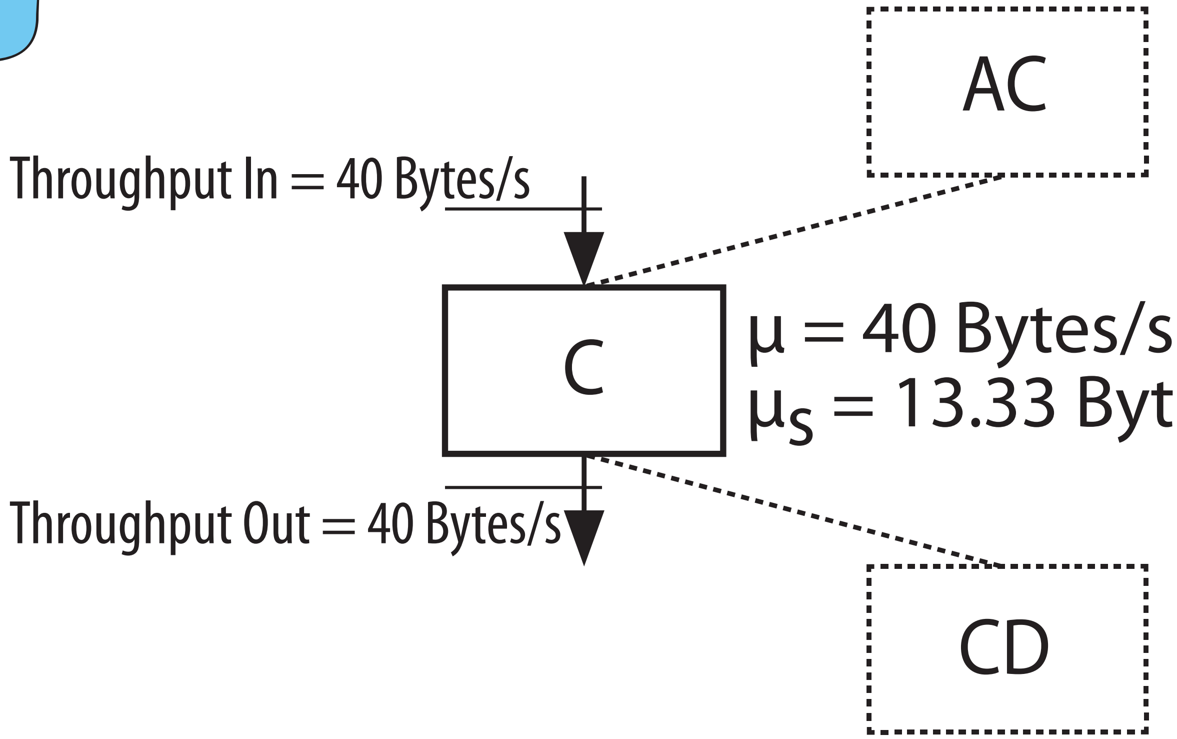
Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Our Example







μ = 40 Bytes/s

μ_S = 13.33 Bytes/s

Routing Fraction (F_r)=1.0

Gain Function (γ)=1.0

Expected Departure Rate (E_D)=13.33 Bytes/s

Flow Model

Conservation of Flow

$$\sum_{j|(i,j)\in E_F} f(\overrightarrow{V_iV_j}) - \sum_{j|(j,i)\in E_F} f(\overrightarrow{V_jV_i}) = \begin{cases} + & i = s \\ 0 & i = \text{circulation} \\ - & i = t \end{cases}$$

Edge Capacity Constraint

$$f(\overrightarrow{V_iV_j}) \leq C(\overrightarrow{V_iV_j})$$

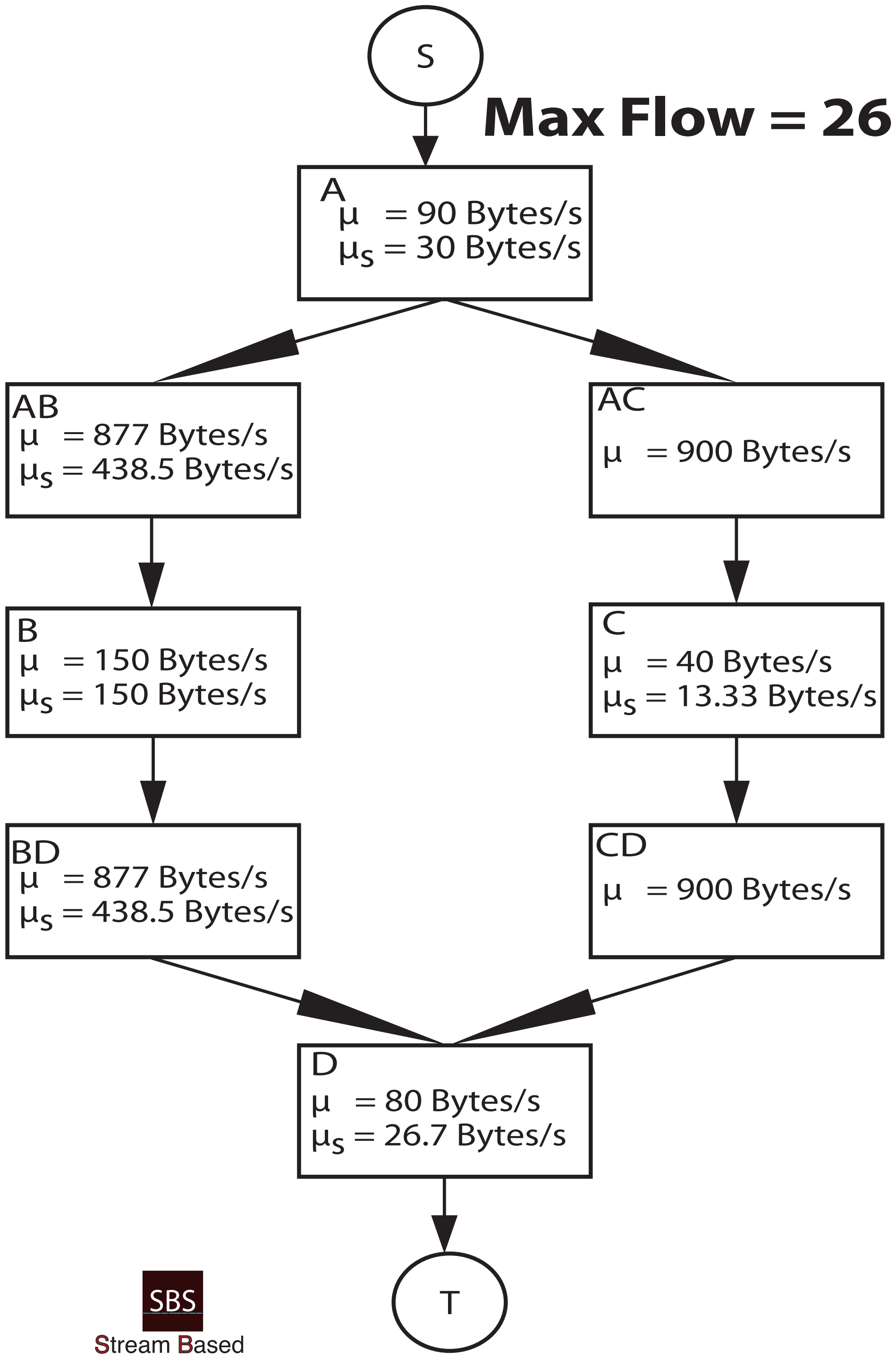
Routing Constraint

$$\frac{f(\overrightarrow{V_iV_j})}{\sum_{x=1}^{N} f(\overrightarrow{V_iV_x})} = R(\overrightarrow{V_iV_j})$$

SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St. Louis

Example finished



Max Flow = 26.7 B/s

S

A
μ = 90 Bytes/s
μ_S = 30 Bytes/s

AB
μ = 877 Bytes/s
μ_S = 438.5 Bytes/s

AC
μ = 900 Bytes/s

B
μ = 150 Bytes/s
μ_S = 150 Bytes/s

C
μ = 40 Bytes/s
μ_S = 13.33 Bytes/s

BD
μ = 877 Bytes/s
μ_S = 438.5 Bytes/s

CD
μ = 900 Bytes/s

D
μ = 80 Bytes/s
μ_S = 26.7 Bytes/s

T

Steps Recap:

- Start with a mapped application topology
- Parameterize the model
- Set the edge capacity equal to the expected departure rate
- Solve for maximum flow

Testing Methodology

- Test the model on multiple real applications (**JPEG encode, DES encrypt**).
- **Generate random synthetic applications** to explore a wider range of application topologies.
- **Randomly map applications** to available hardware using uniform random process.
- **Measure throughput and queue occupancy on** generated Application / Hardware mappings at **each stream (edge)**.

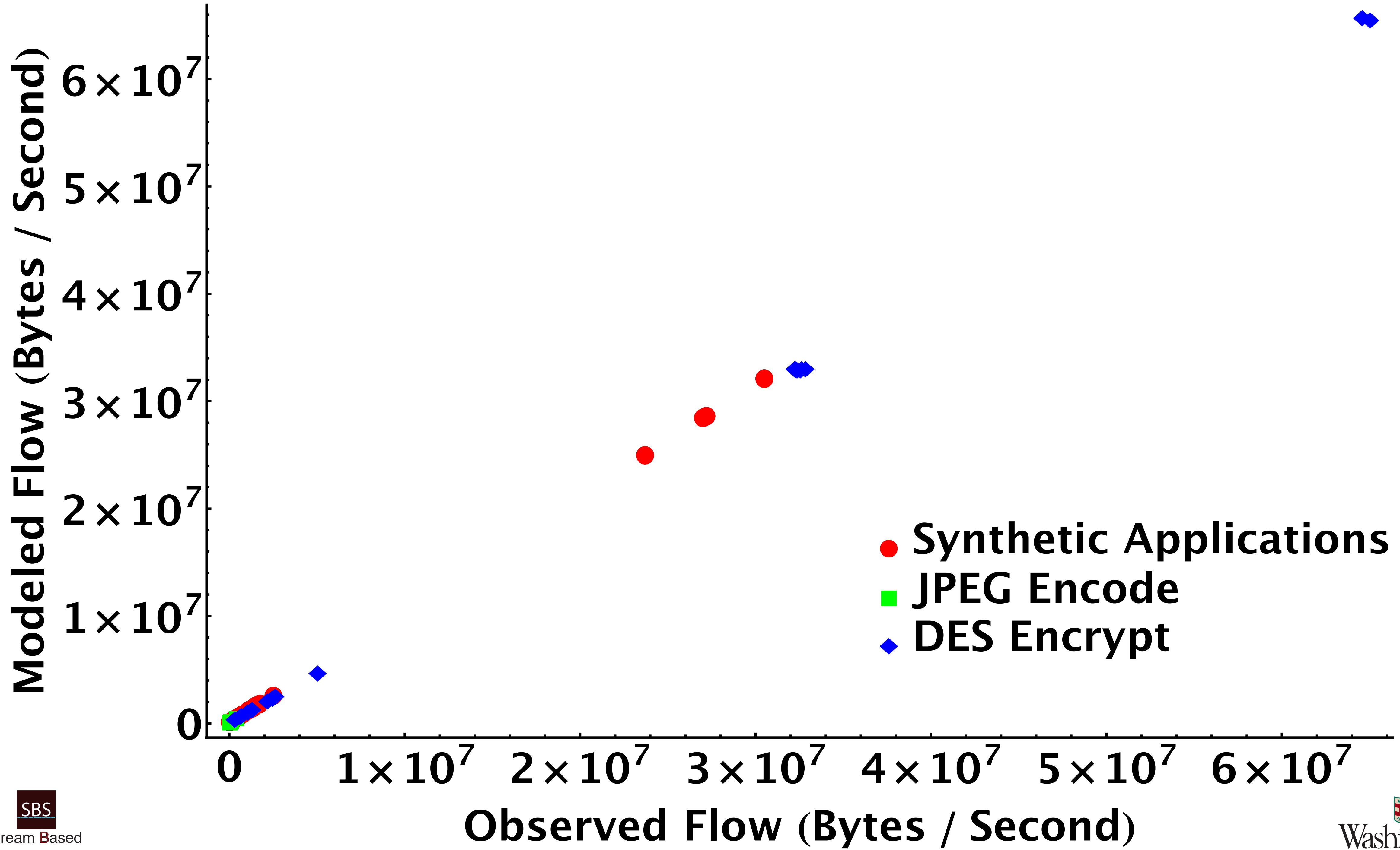
Synthetic Application Stats

Statistic	Mean	Std. Deviation
Number of Vertices	21	17.52
Kernels per Resource	3.6	3.51
Gain or Loss	0.98	1.03
Routing Probability (F_r)	0.585	0.340
Service Time (μ)	Varies, mean 20 μs	
Packet Size	Varies, 16-Bit to 64-Bit	
Implementations	Hardware and Software	

Utilized Hardware

Specification	Machine 1 (x 2)	Machine 2
CPU	12 x 2.4 GHz AMD Opteron	4 x 3.1 GHz Intel Xeon E3
FPGA	2 x Virtex-4 LX 100	None
RAM	32GB DDR2	8GB DDR3

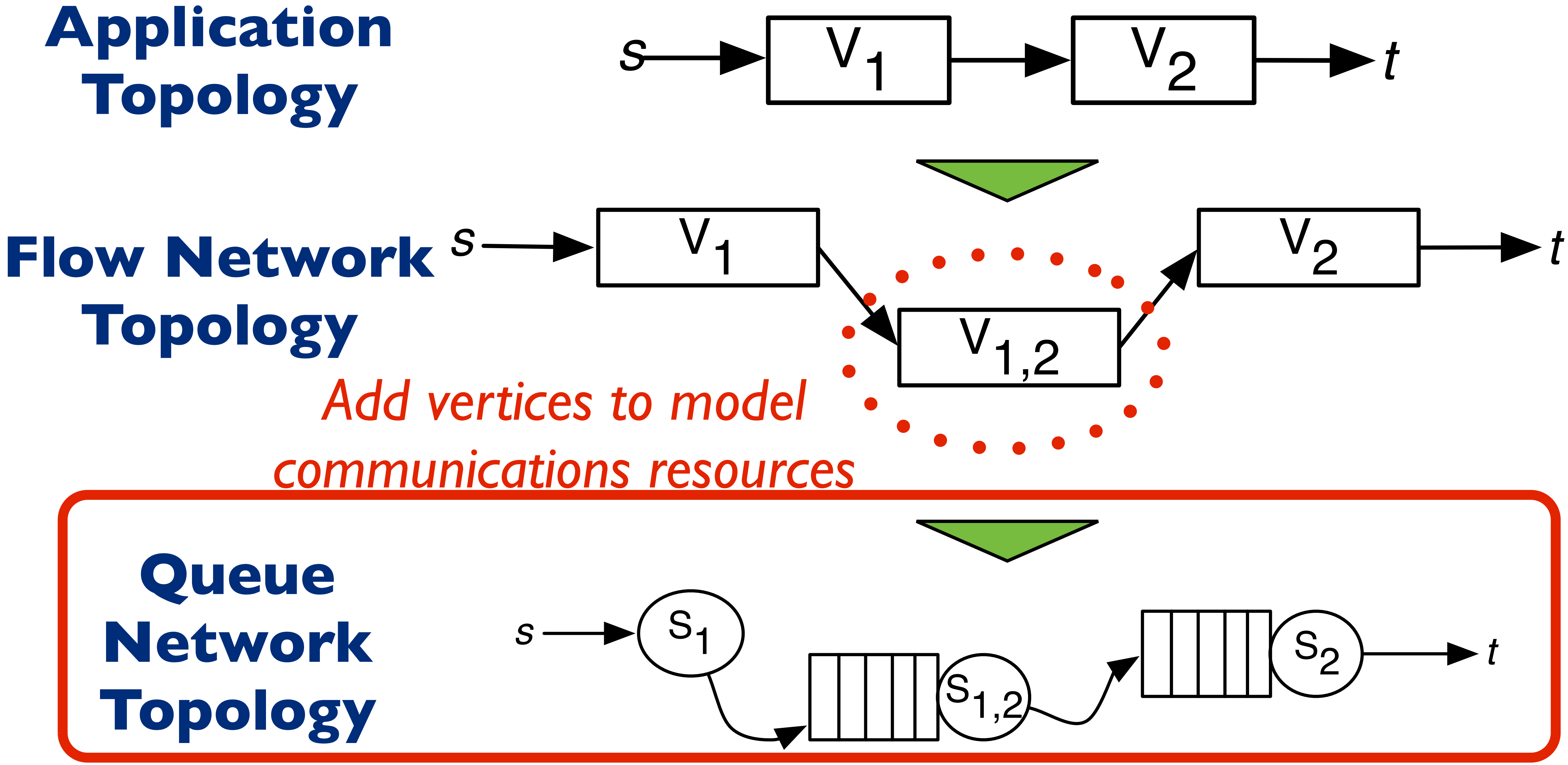
Flow Model Results



SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St. Louis

Overall Model Layout



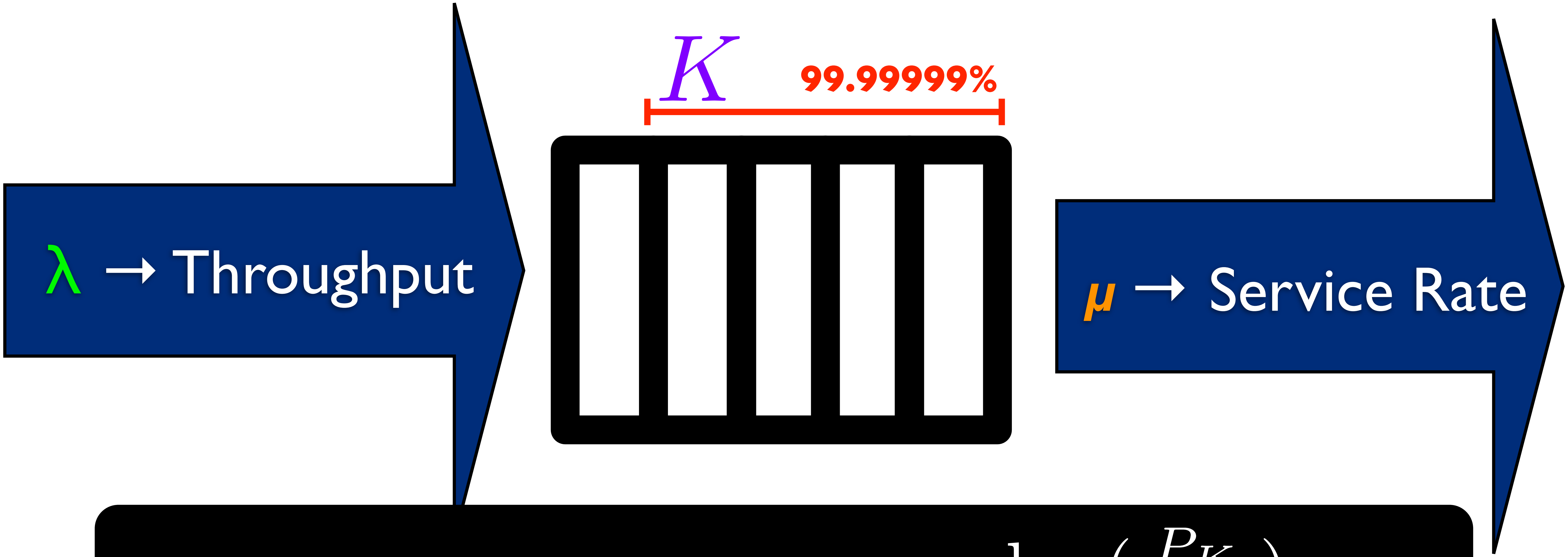
Application Topology

Flow Network Topology

Add vertices to model communications resources

Queue Network Topology

M/M/1 Occupancy Model



$$K[\rho = \frac{\lambda}{\mu}, P_K = 10^{-7}] = \frac{\log(\frac{P_K}{1-\rho})}{\log(\rho)} - 1$$

Queue Model Results

Step 1:

Modeled Occupancy	Observed Occupancy	Percent Error
M	O	(*M* - O / O)

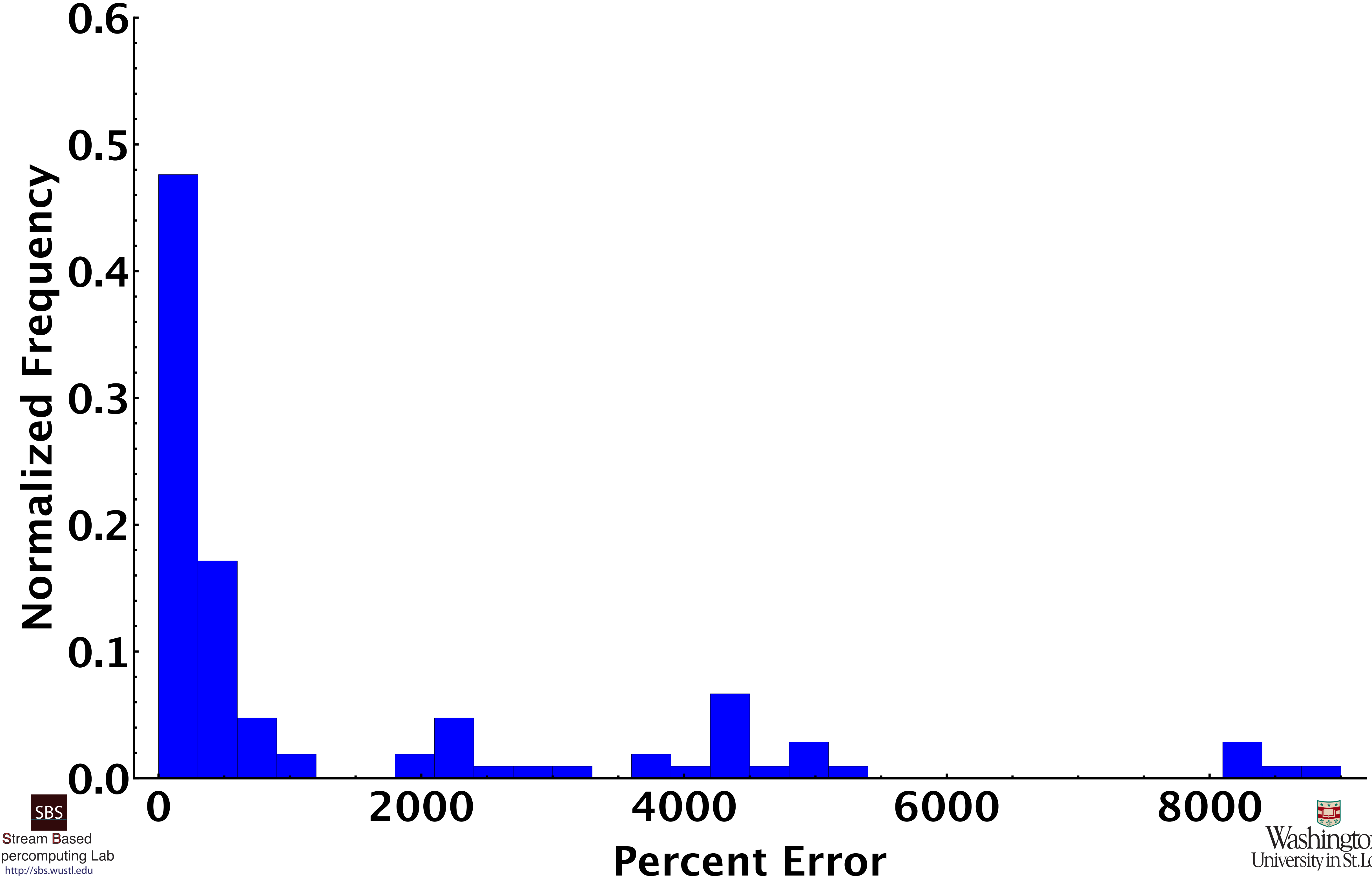
Step 2: Combine DES Encrypt, JPEG Encode and Synthetic Applications

Step 3: Make a histogram

SBS
Stream Based Supercomputing Lab
http://sbs.wustl.edu

Washington University in St.Louis

Queue Model Results



SBS
Stream Based
Supercomputing Lab
http://sbs.wustl.edu

Washington
University in St. Louis

Conclusion

- Showed that a generalized maximum flow model can be used to solve for max flow of a queueing network.
- Demonstrated the flow model is reliable on real systems
- Simple M/M/1 queueing model is insufficient to estimate buffering requirements

References

Slides and Software

@

http://sbs.wustl.edu