



**Efficient Buffer Capacity and Scheduler Setting Computation
for
Soft Real-Time Stream Processing Applications**

Marco Bekooij, Maarten Wiggers, and Jef van Meerbergen



Outline

- ▶ Context: real-time stream processing
- ▶ Problem statement
- ▶ Cyclo-static dataflow representation of an execution trace
- ▶ Low complexity FIFO capacity computation algorithm
- ▶ Results of artificial test-cases
- ▶ Results of an H263 video decoder case-study
- ▶ Conclusion

Context

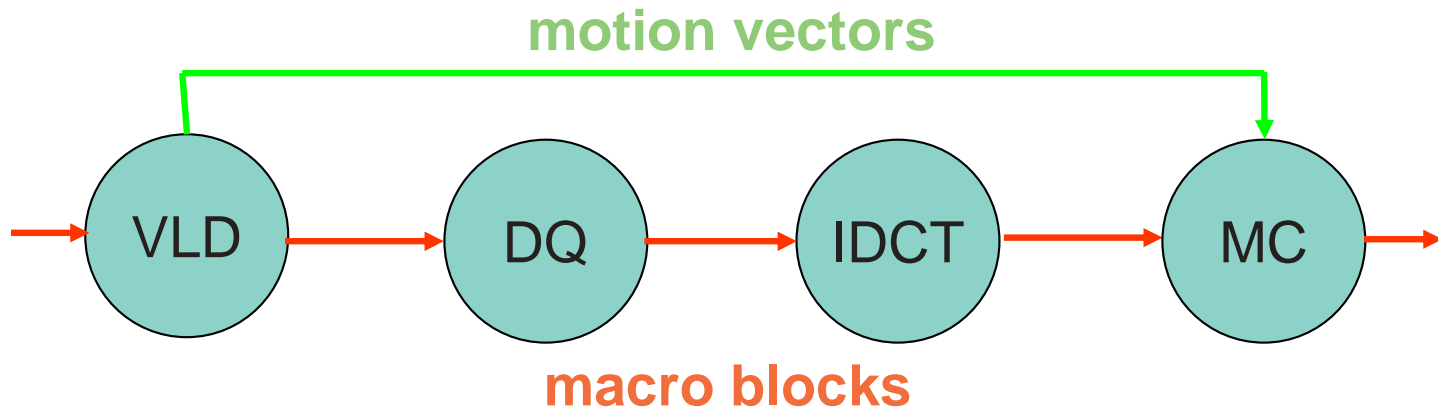
- ▶ Real-time stream processing on embedded multiprocessor systems
 - Smart phones
 - Car-radios
- ▶ Real-time requirements
 - Hard real-time = guarantee for all streams the throughput and latency
 - Use of worst-case execution times
 - Input and output behavior of tasks known at design time
 - Low resource utilization if variation in execution time of the tasks is high
 - Soft real-time = throughput and latency target
 - Execution times + correlation
 - Data dependent input/output behavior of tasks
 - ***Improved resource utilization***
 - Throughput and latency is not guaranteed for every possible stream
 - Deadlock freedom cannot be guaranteed for every stream

Related work

- ▶ Throughput analysis for soft real-time applications
 - Simulation at different abstraction levels (CP-T, PV-T, cycle-true)
 - Trade-off between run-time and accuracy
 - Probabilistic analysis (Markov chain, Markov decision process)
 - Long running averages
 - Approximation of correlation to reduce run-time analysis algorithm
 - Undefined accuracy for short time-intervals
- ▶ This work: throughput synthesis for soft real-time applications
 - Compute FIFO capacities given throughput constraint
 - Assuming a predictable multiprocessor system
 - For one input stream, it is guaranteed that throughput constraint is met

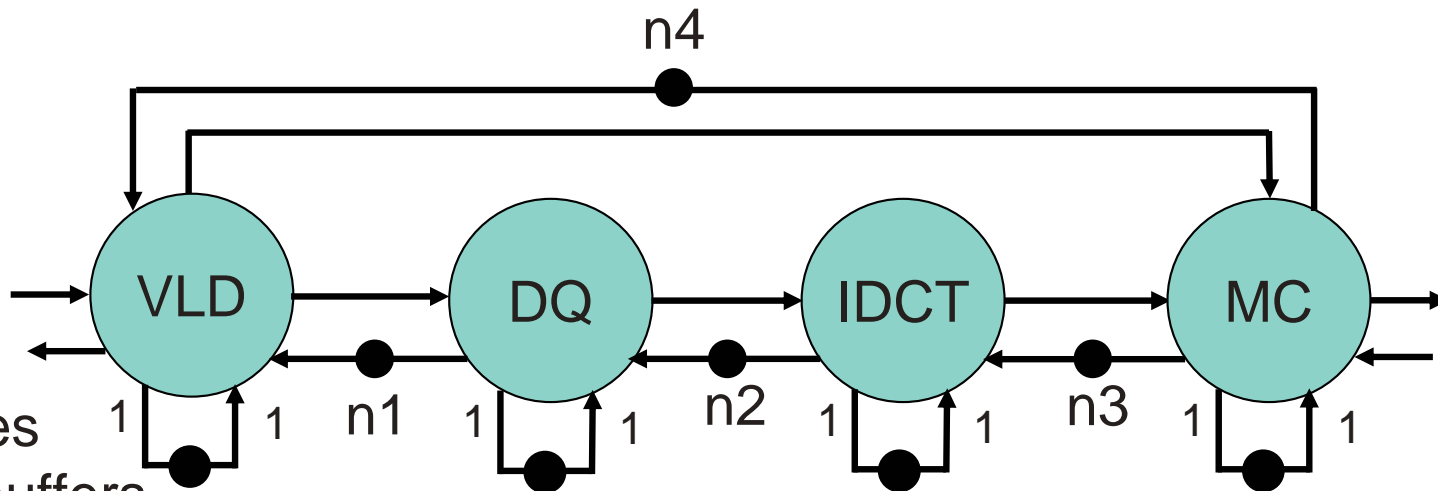
Process network: H263 video decoder

YAPI
Process
Network



No firing rules
Unbounded FIFO buffers
Untimed

Dynamic
Dataflow (DDF)



Implicit firing rules
Bounded FIFO buffers
Timed

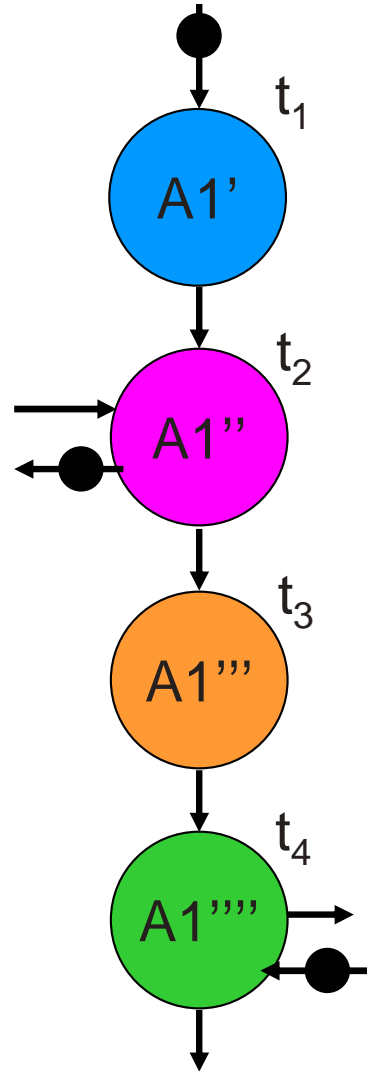
Cyclo-static dataflow representation of a trace

YAPI process

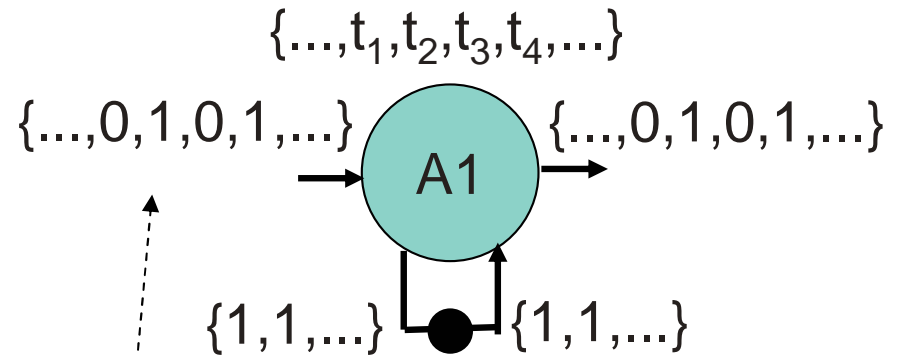
```

while(1){
  ...
  read(FIFO1,x);
  ...
  if(x>0){
    write(FIFO2,z);
  }
}
    
```

trace



Cyclo-static dataflow actor



- Firing rules explicit
- Correlation fully captured

Input of buffer computation algorithm!

Buffer capacity computation

Run-time(number_of_phases)

| phases | 10^2 | 10^3 | 10^4 | 10^5 | 10^6 | 10^7 |
|--------|--------|--------|--------|--------|--------|--------|
| exact | 0.05s | 2.09s | 218s | >2h | >2h | >2h |
| approx | 0.02s | 0.09s | 0.79s | 7.7s | 78s | 780s |

Linear complexity in the number of phases: $O(|V|^4 + |V| * |E| * \text{phases})$

Processor sharing (time division multiplex)

f = phase

v_i = i -th actor

p = time division multiplex period

s_i = i -th slice length

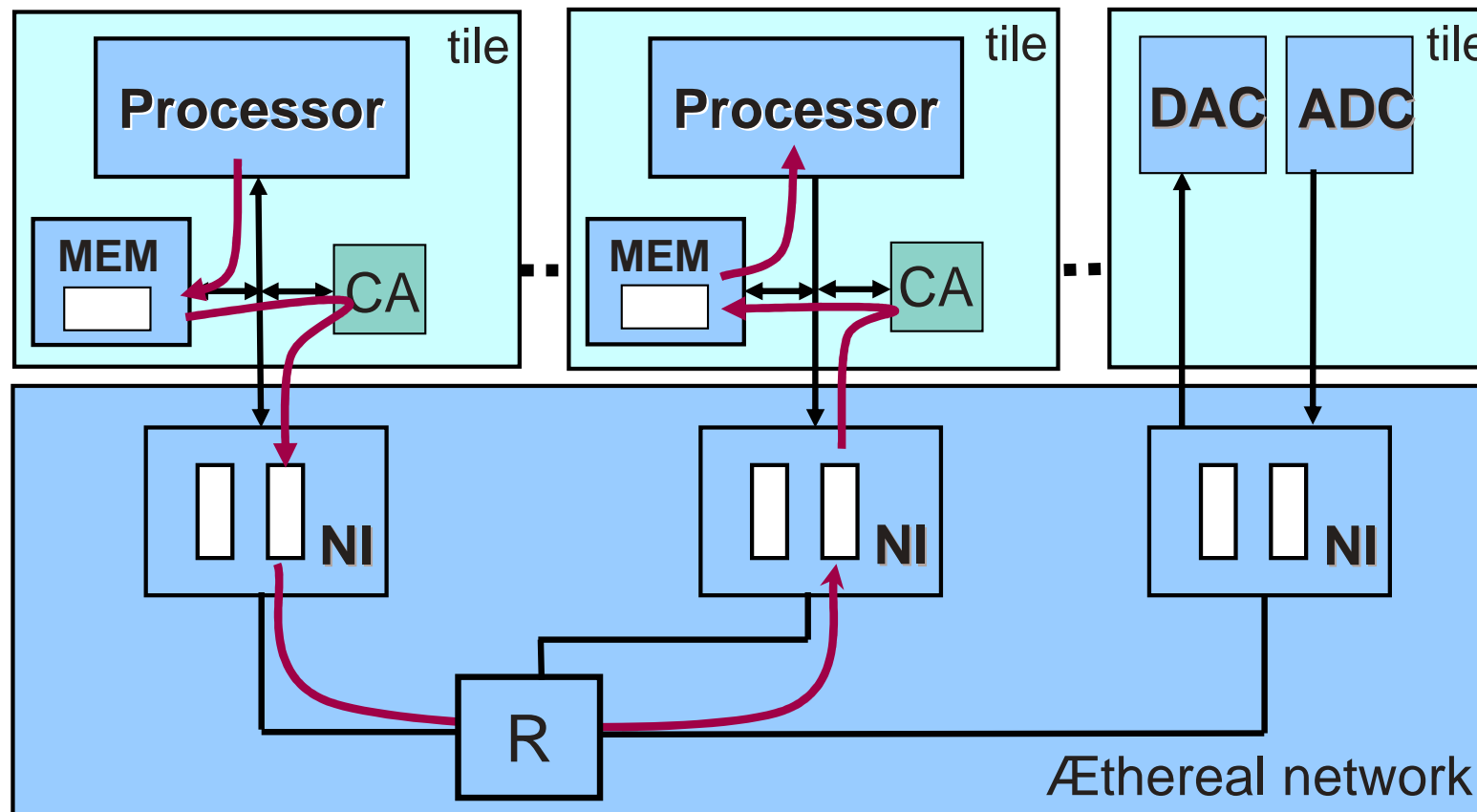
$\phi(v_i, f)$ = execution time with stalls

$\rho(v_i, f)$ = response time

$$\rho(v_i, f) = \phi(v_i, f) + (p - s_i) \lceil \phi(v_i, f) / s_i \rceil$$

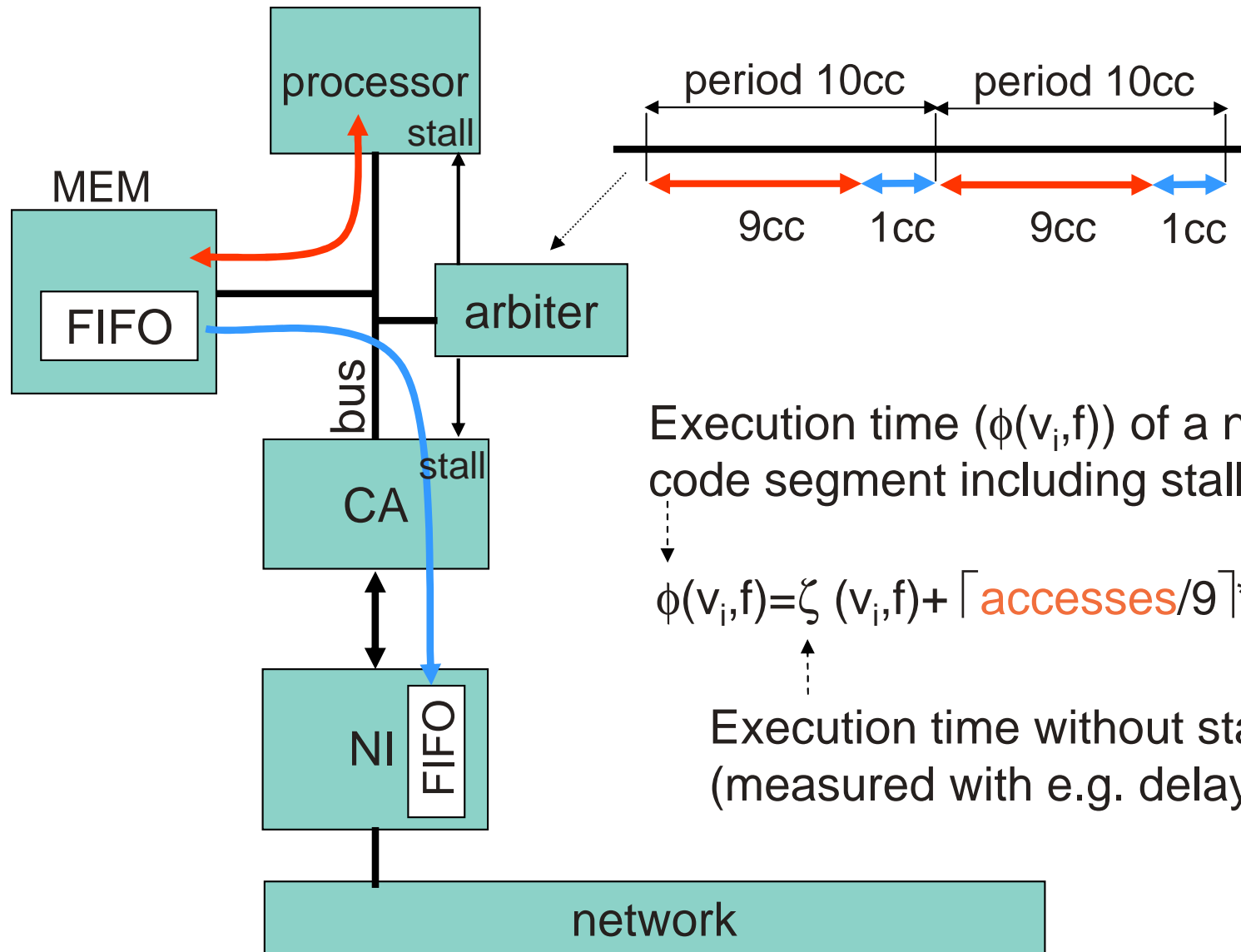
No sharing then **response time = execution time**

Predictable multiprocessor architecture



- Time division multiplex (TDM) arbitration for every shared resource
- Guaranteed throughput connections in network
 - guaranteed min bandwidth, max latency, in-order delivery, flow control
- A processor can only access the memory in its tile
 - communication assist copies data between network interface FIFO and MEM

Conservative execution time estimate

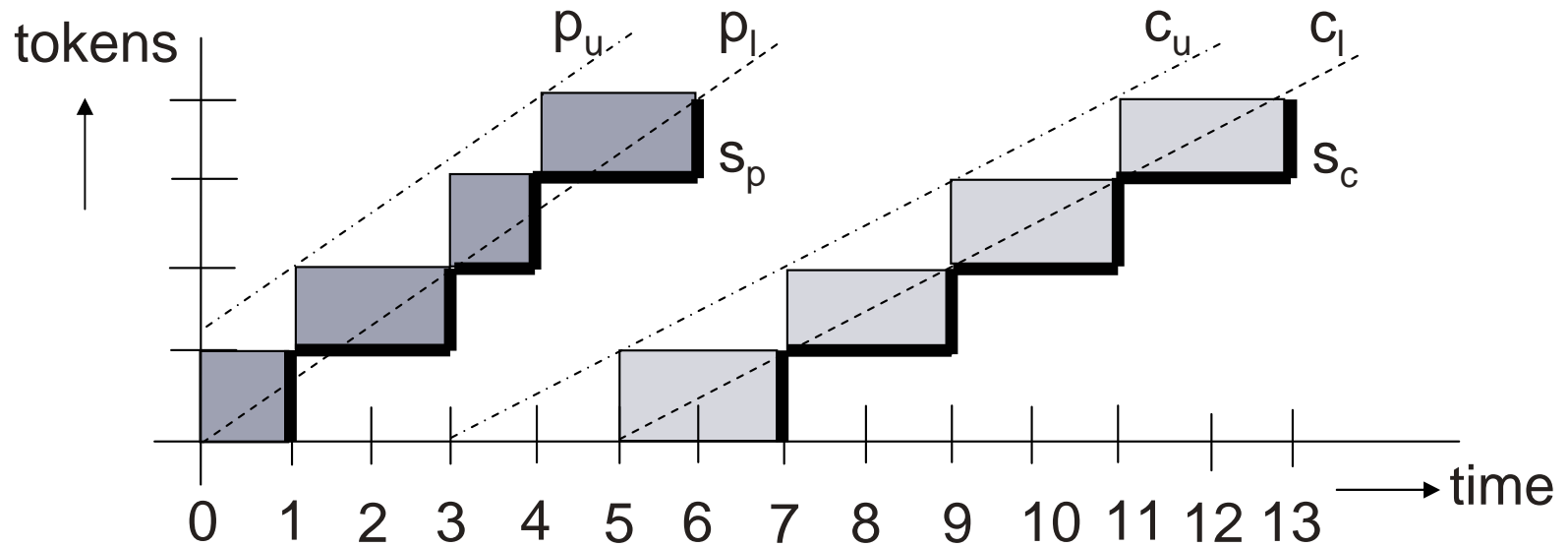
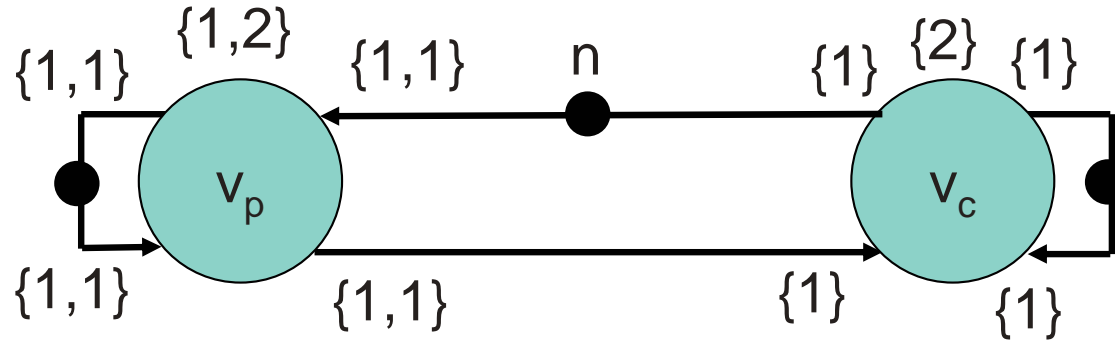


Execution time ($\phi(v_i, f)$) of a non-blocking code segment including stall cycles

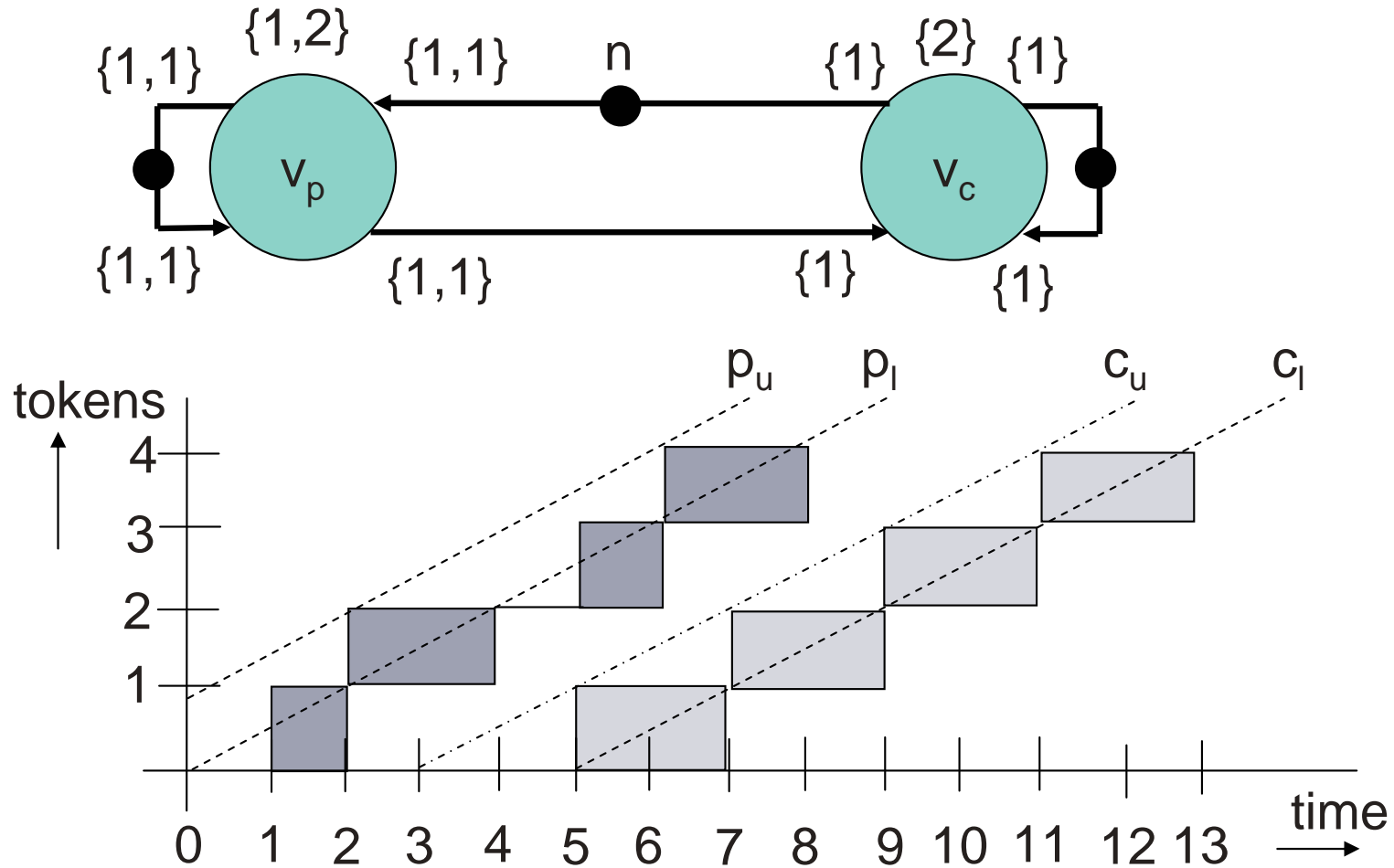
$$\phi(v_i, f) = \zeta(v_i, f) + \lceil \text{accesses} / 9 \rceil * 10$$

Execution time without stalls
(measured with e.g. delay statements)

FIFO capacity computation example

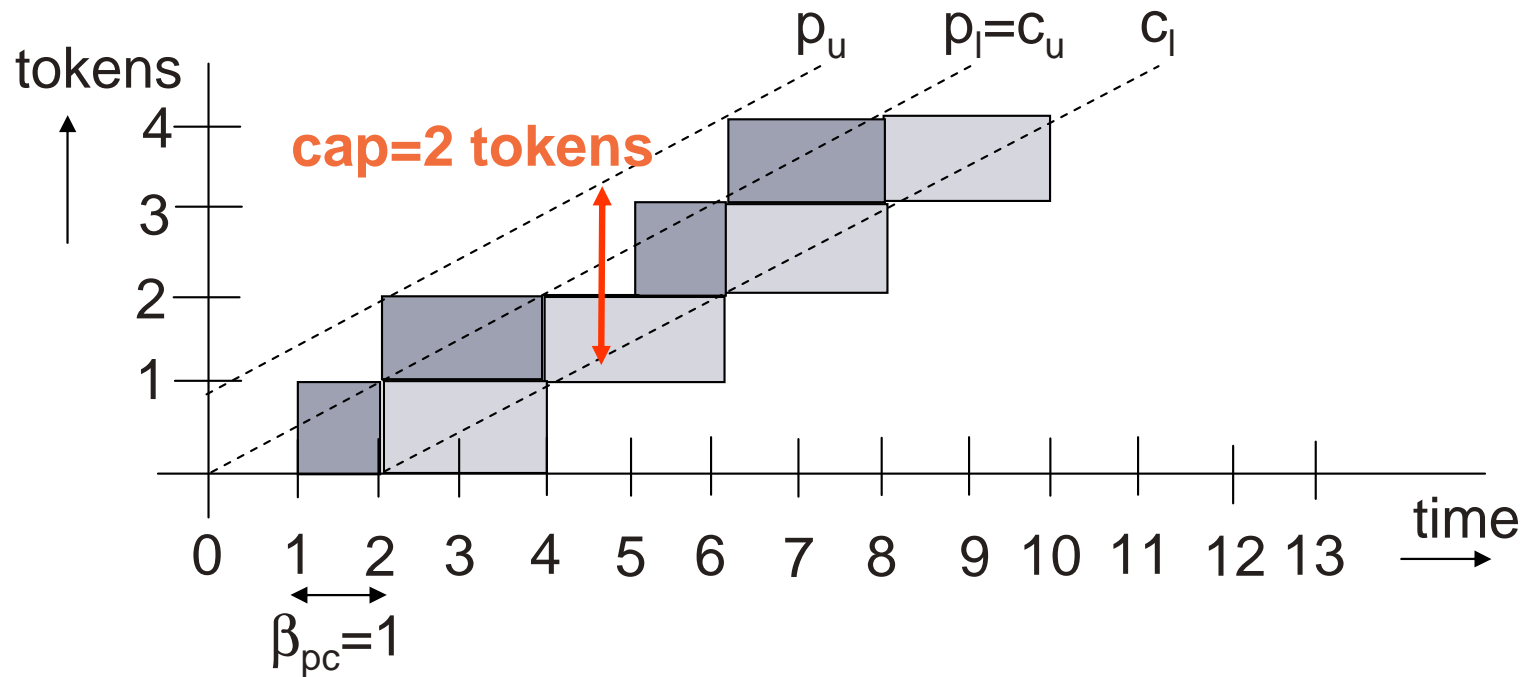
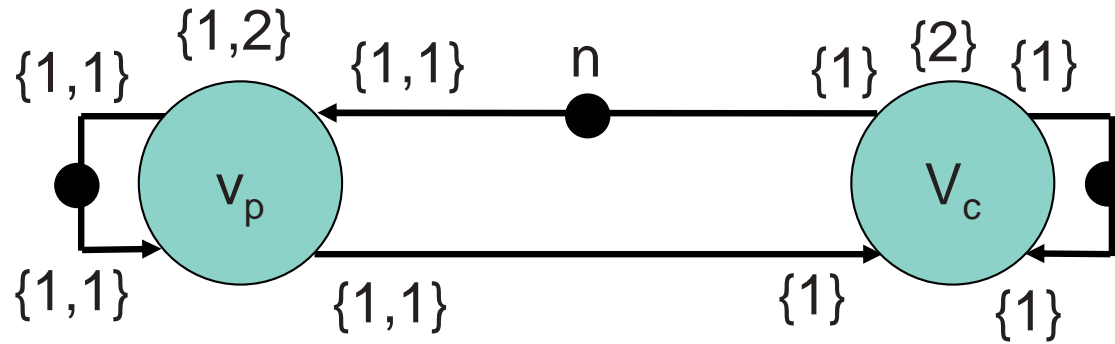


Stretched schedule

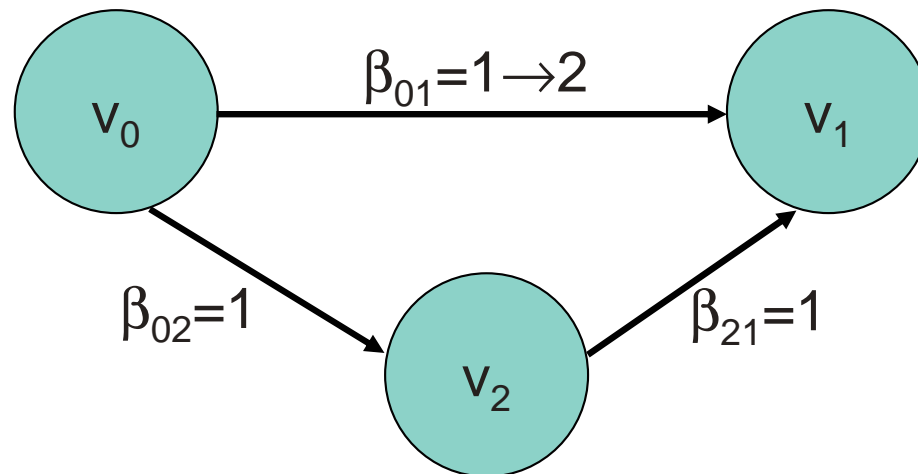
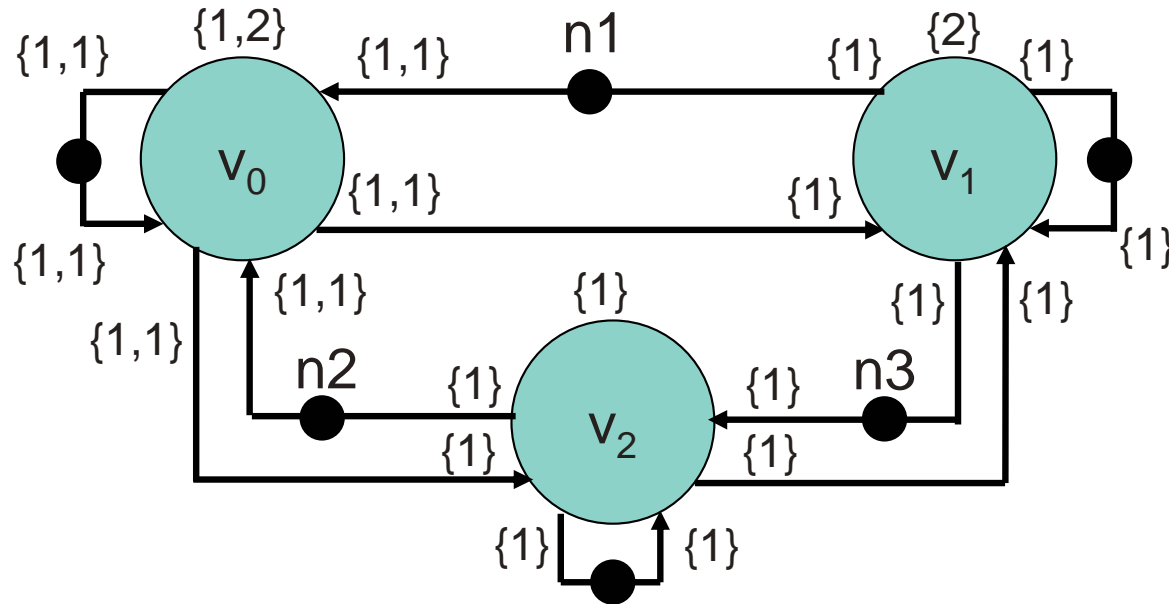


Monotonic execution: earlier production \Rightarrow earlier enable \Rightarrow earlier production

Stretched schedule

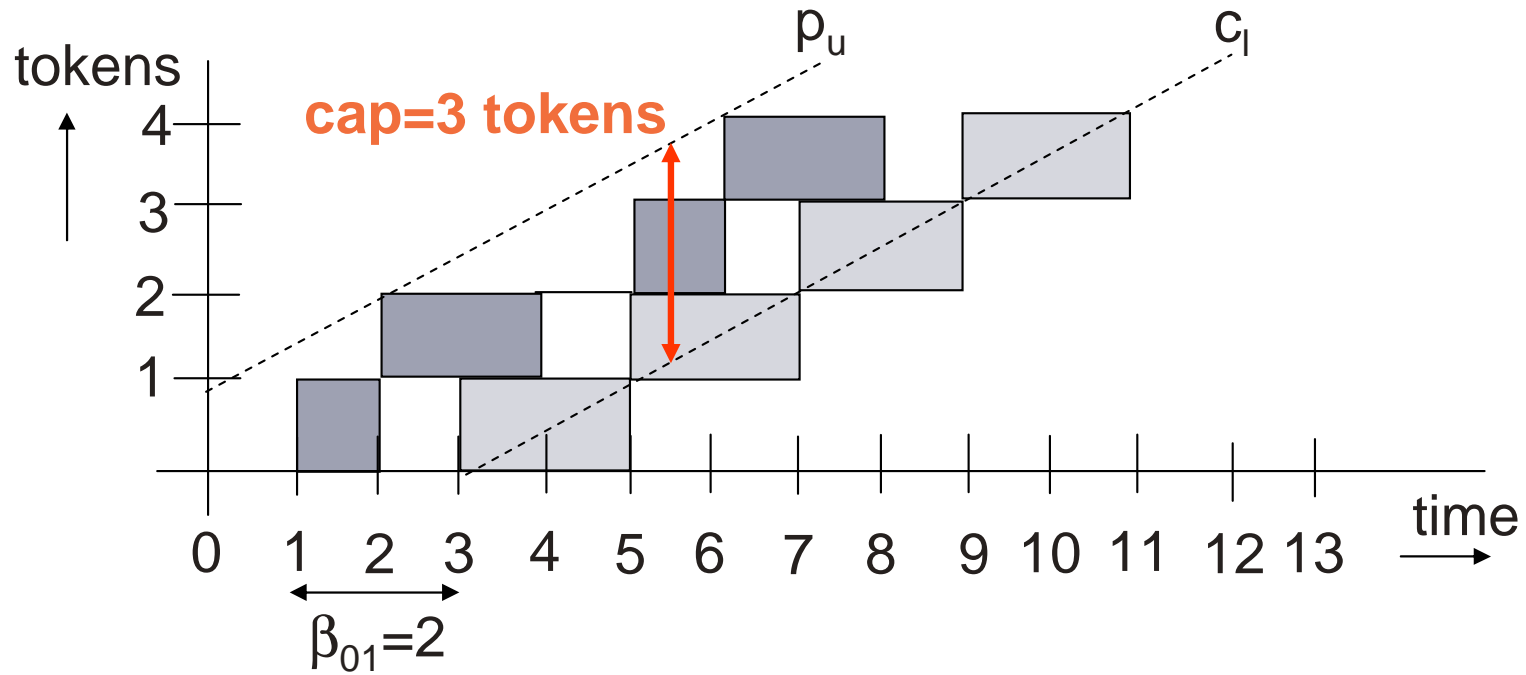


Multiple distance constraints

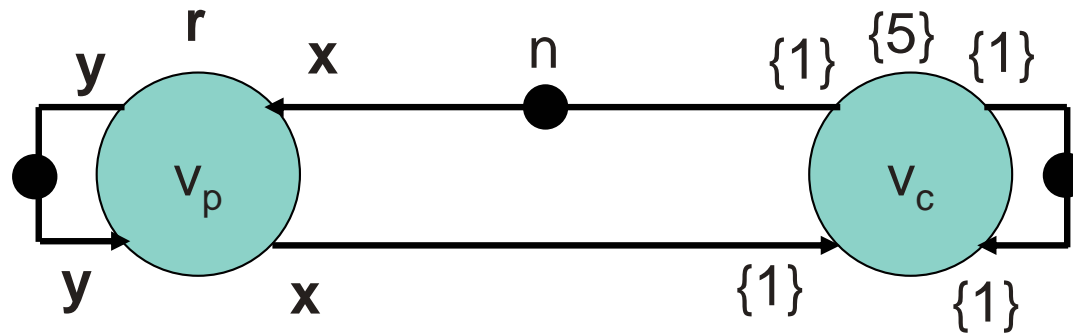


min-cost max-flow

Capacity buffer $v_0 \rightarrow v_1$



Artificial test-case: varying response-time



$$\mathbf{x} = \{1, 1, \dots, 1\}$$

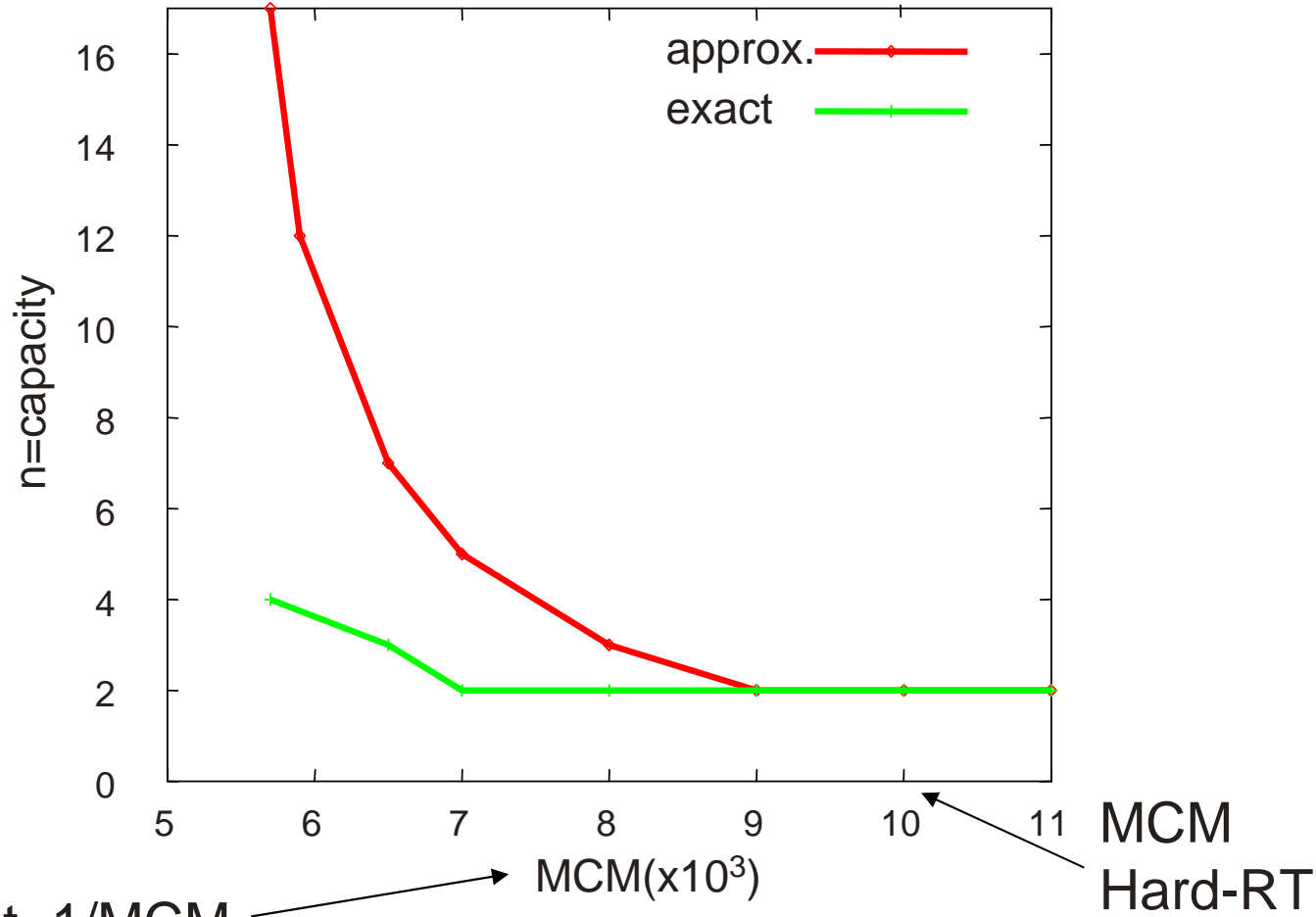
$$\mathbf{y} = \{1, 1, \dots, 1\}$$

$$\mathbf{r} = \{10, 4, \dots, 9\}$$

$$\theta(v_p) = 1000$$

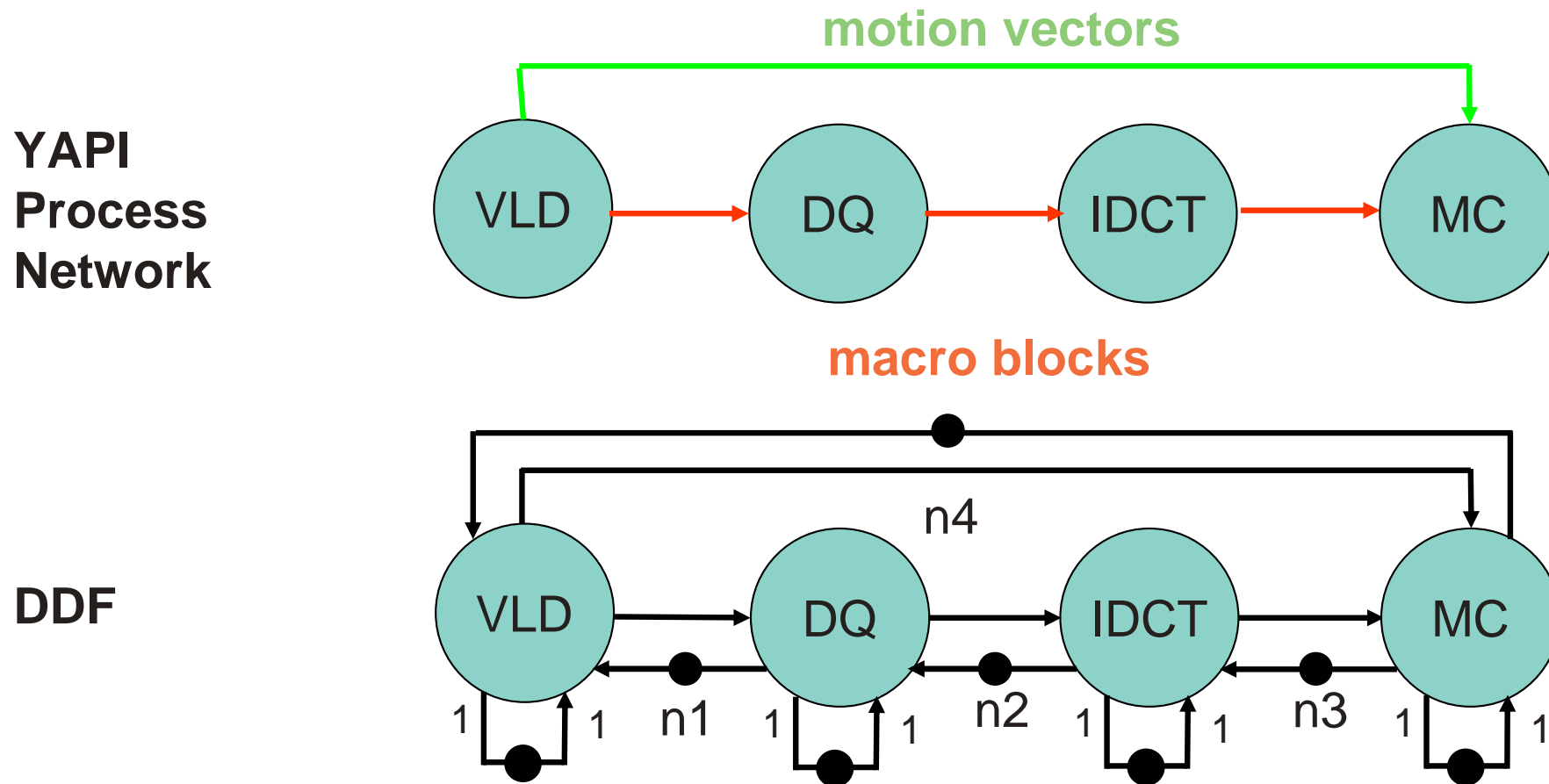
$1 \leq x_i \leq 10$, uniform distribution

Varying response-time: capacity(MCM)



Similar for varying input output behavior

H263 video decoder case-study



Communication latency network was not taken into account

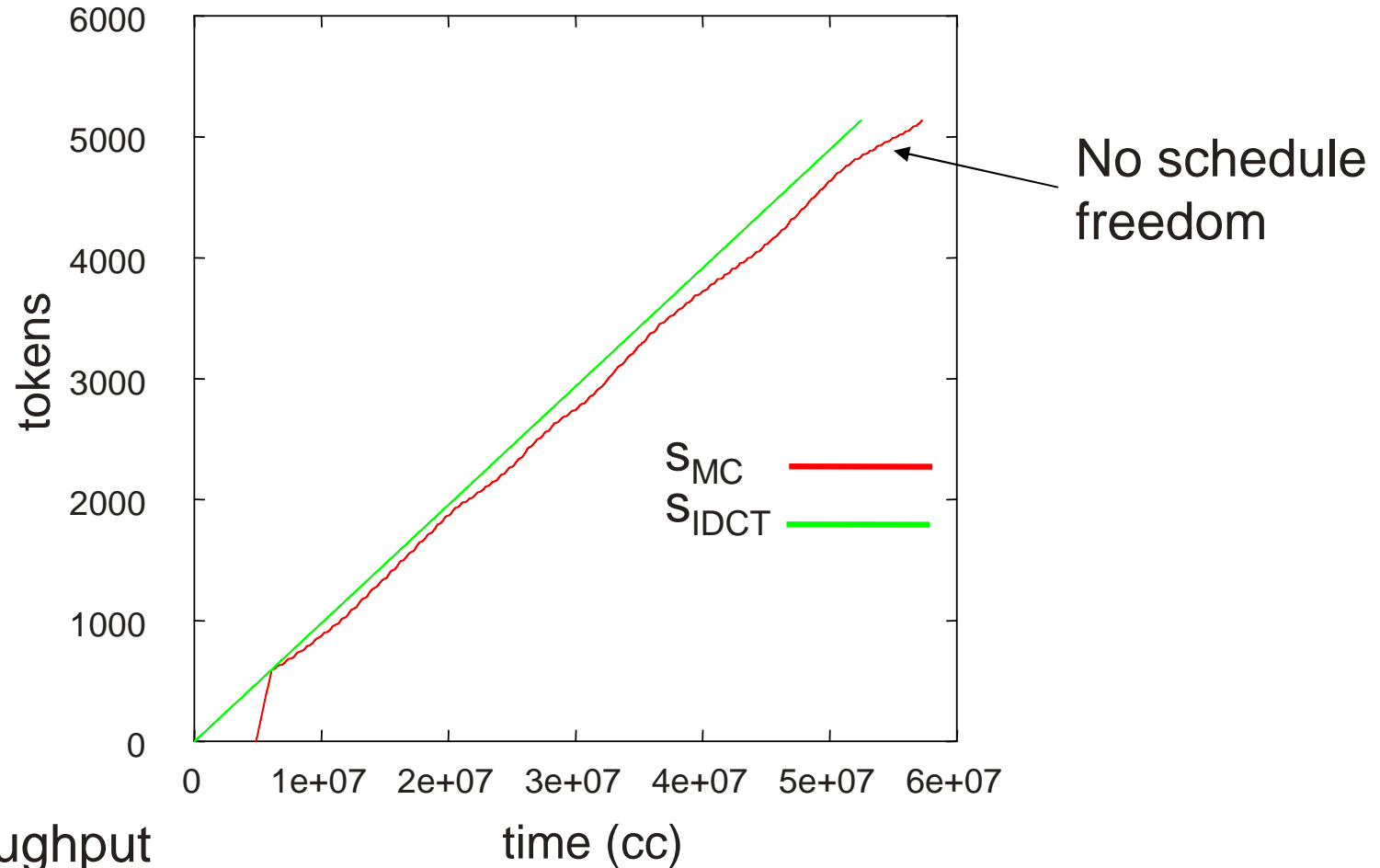
FIFO_capacity(desired_MCM)

| Desired MCM ($52 \cdot 10^6$) | n1 VLD→DQ | n2 DQ →IDCT | n3 IDCT →MC | n4 VLD →MC |
|------------------------------------|--------------|----------------|----------------|---------------|
| 10 | 3 | 2 | 8 | 86 |
| 5 | 3 | 2 | 8 | 85 |
| 2 | 5 | 2 | 12 | 78 |
| 1.5 | 6 | 2 | 43 | 166 |
| 1 | 8 | 2 | 468 | 984 |

Phases = 143010 phases, run-time = 14 sec



Linear bounds not tight \Rightarrow large buffer



Maximum throughput
Desired MCM=52 *10⁶

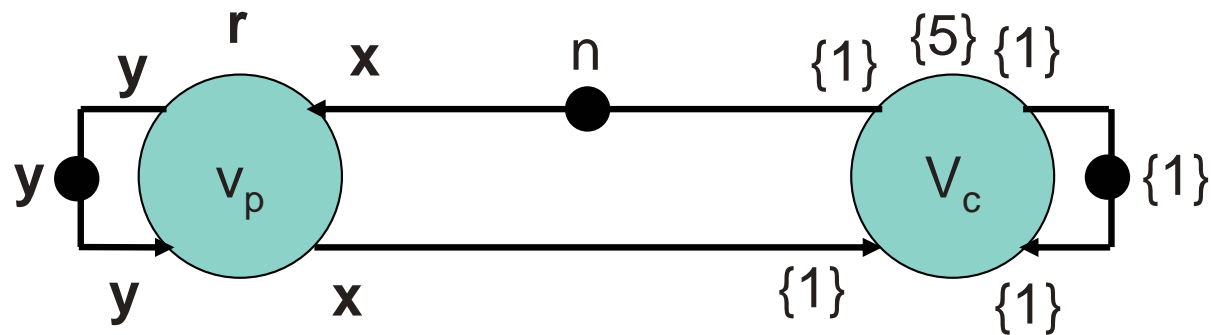
Conclusion

- ▶ Presented a buffer capacity computation technique for soft real-time applications that are described as YAPI process networks
 - Given a throughput constraint
 - Conservative capacities for one stream \Rightarrow correlation is fully taken into account
- ▶ Execution trace is encoded in phases of CSDF actors
 - Potentially a large number of phases
 - Low complexity algorithm for computation of conservative buffer capacity estimates
 - Less tight results if desired throughput gets closer to maximum throughput
- ▶ H263 video decoder case-study
 - Larger buffer if desired MCM \rightarrow minimum MCM
 - Run-time was 14 seconds for a CSDF with in total 143010 phases

Questions?



Artificial test-case: varying rates



$$\mathbf{y} = \{1, 1, \dots, 1\}$$

$$\mathbf{r} = \{1, 1, \dots, 1\}$$

$$\mathbf{x} = \{2, 4, \dots, 1\}$$

$$\theta(v_p) = 1000$$

$1 \leq x_i \leq 4$, uniform distribution

Varying rates: capacity (MCM)

