Void Creation: Reculer pour mieux sauter

Kurt Van Hautegem, Wouter Rogiest & Herwig Bruneel

{kurt.vanhautegem ✉, wouter.rogiest, herwig.bruneel}@telin.UGent.be

Ghent University, Belgium (UGent)

Department of Telecommunications and Information Processing (TELIN)

Stochastic Modelling and Analysis of Communication Systems (SMACS)

Montaigne

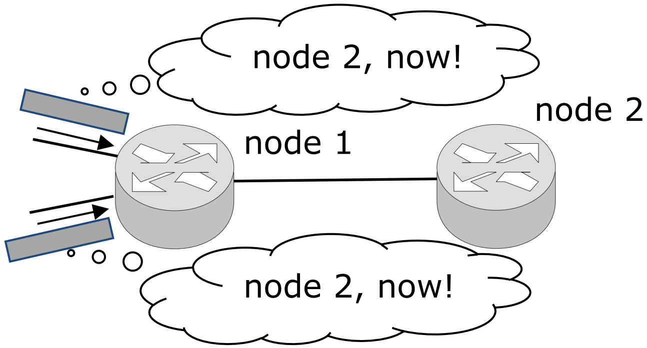
Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

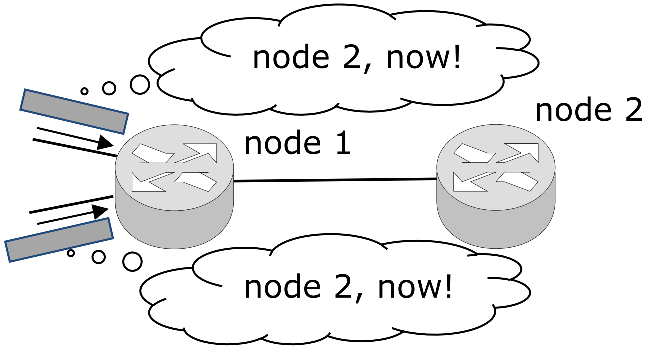
Contention & resolution

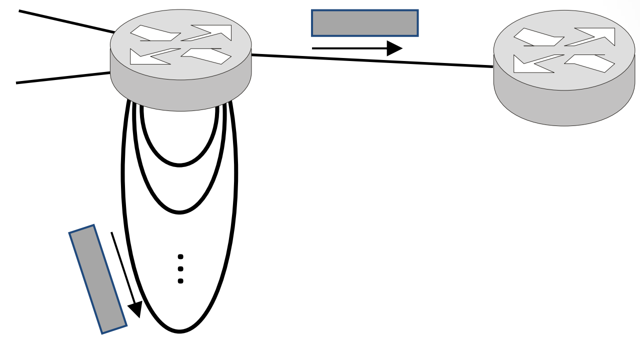


Arrival process

- single wavelength
- variable packet lengths B
- exponentially distributed interarrival times (Poisson)

Contention & resolution





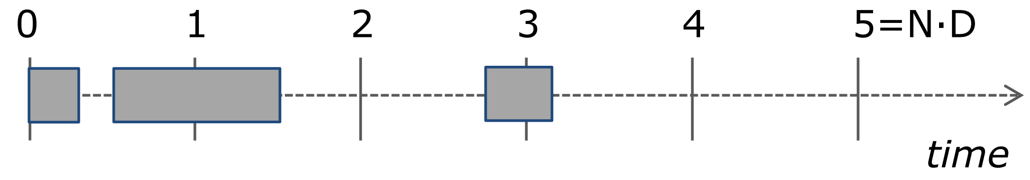
Arrival process

- single wavelength
- variable packet lengths B
- exponentially distributed interarrival times (Poisson)

Fiber Delay Lines (FDLs)

- set of fibers, # = N+1
- lengths j·D, j=0…N
- N= buffer size
- D= granularity = E[B]

Provisional schedule



- shows already scheduled packets upon arrival of a packet
- horizontal axis: future time
- vertical lines: delays of FDLs (N=5, D=1)
- updated at every arrival
- choppy but uniform movement of all packets to the left

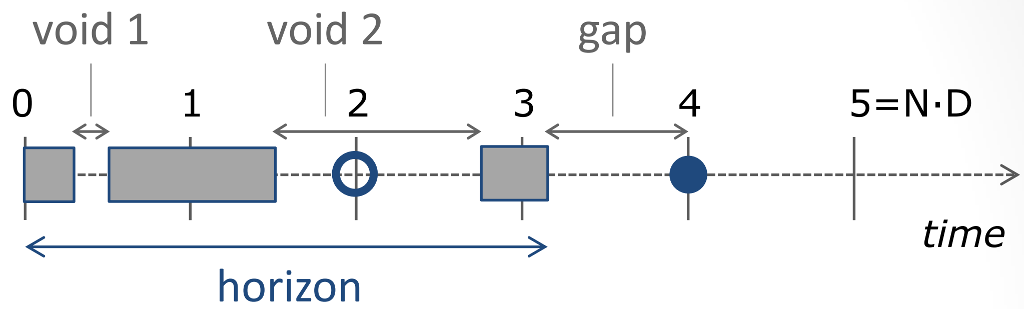
Provisional schedule

choose:

- delay line j (j=0…N)

constraint:

- no overlap



exisiting algorithms

non-void-filling: always first FDL after horizon: ●

- only keep track of horizon

void-filling: fill void if possible: ○ else: ●

- keep track of all voids

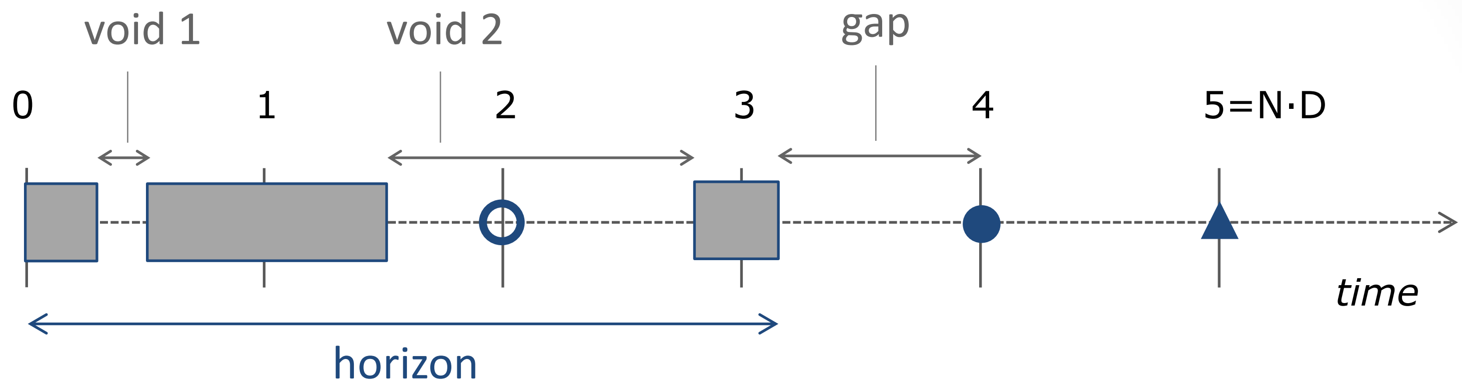
Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Void-creating scheduling algorithm



Always fill a void if possible (○ in example) else choose between:

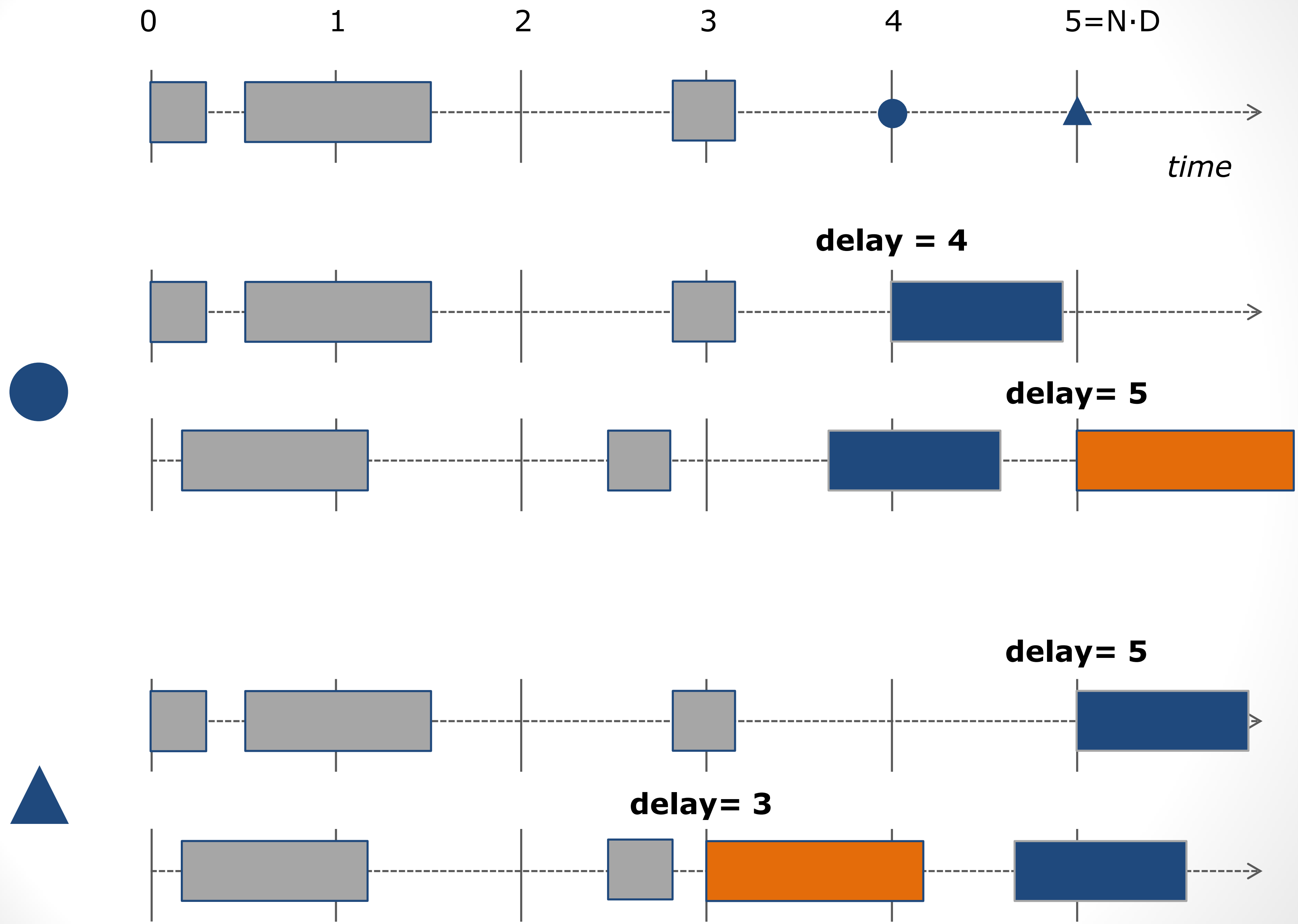
●

- normal scheduling point
- first FDL after horizon
- creates smaller void

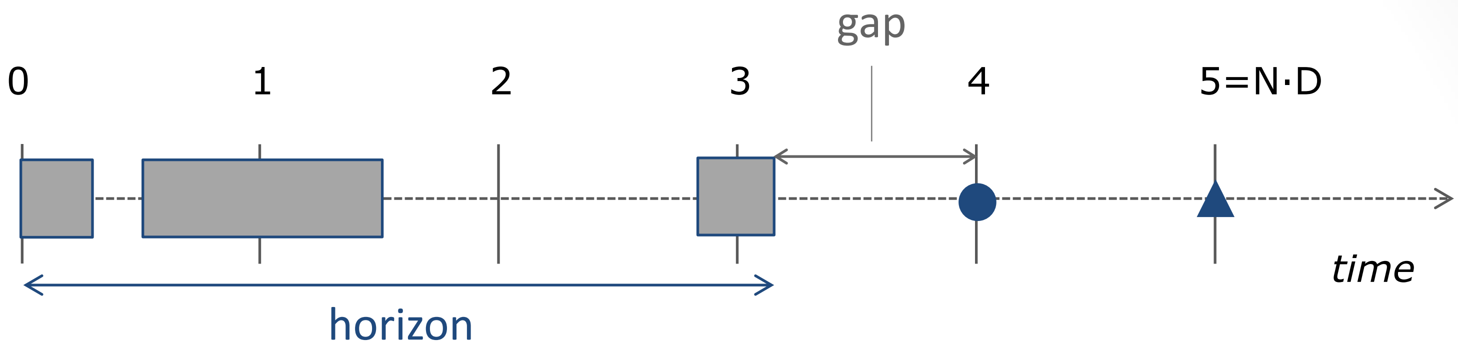
▲

- alternative scheduling point
- second FDL after horizon
- creates larger void

Why ▲ instead of ● ?



Why ▲ instead of ● ?



IF void is filled:

- average delay / packet: ↘

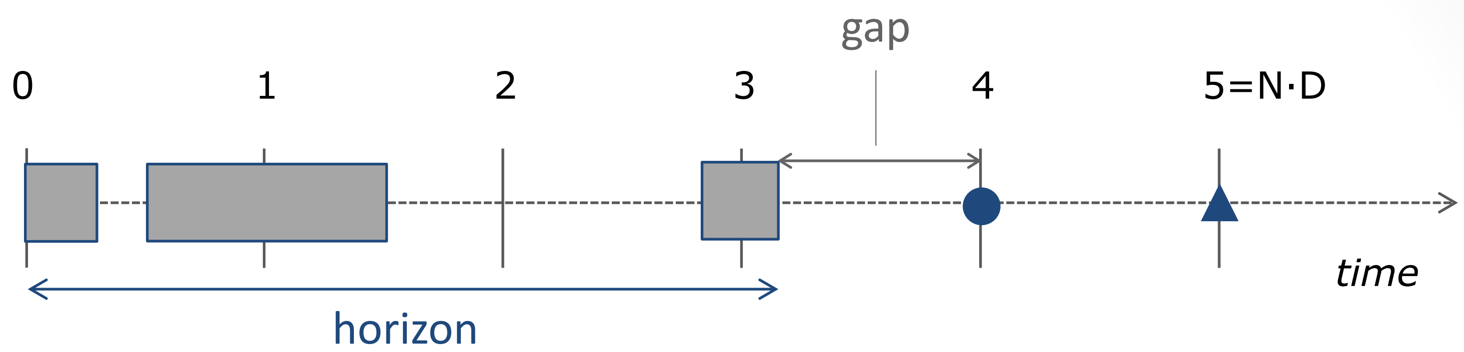
⇩

stacking becomes more dense

⇩

loss probability ↘

Why ▲ instead of ● ?



IF void is filled:

- average delay / packet: ↘

⇩

stacking becomes more dense

⇩

loss probability ↘

- position with respect to FDL has to be favorable
- depends on size arriving packet
- depends on arrival instances future packets (stochastic arrival process)
- larger voids: chance of filling ↗

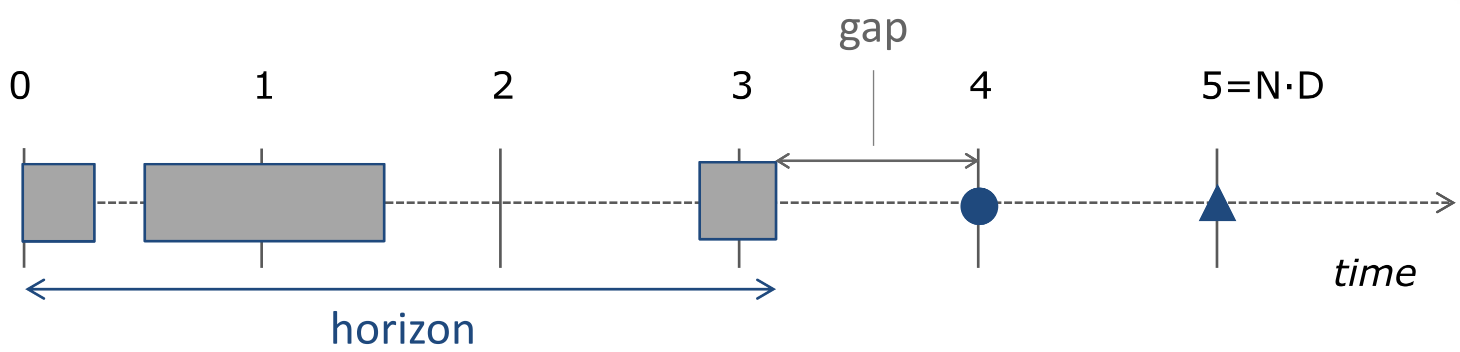
Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Choosing between ▲ and ●



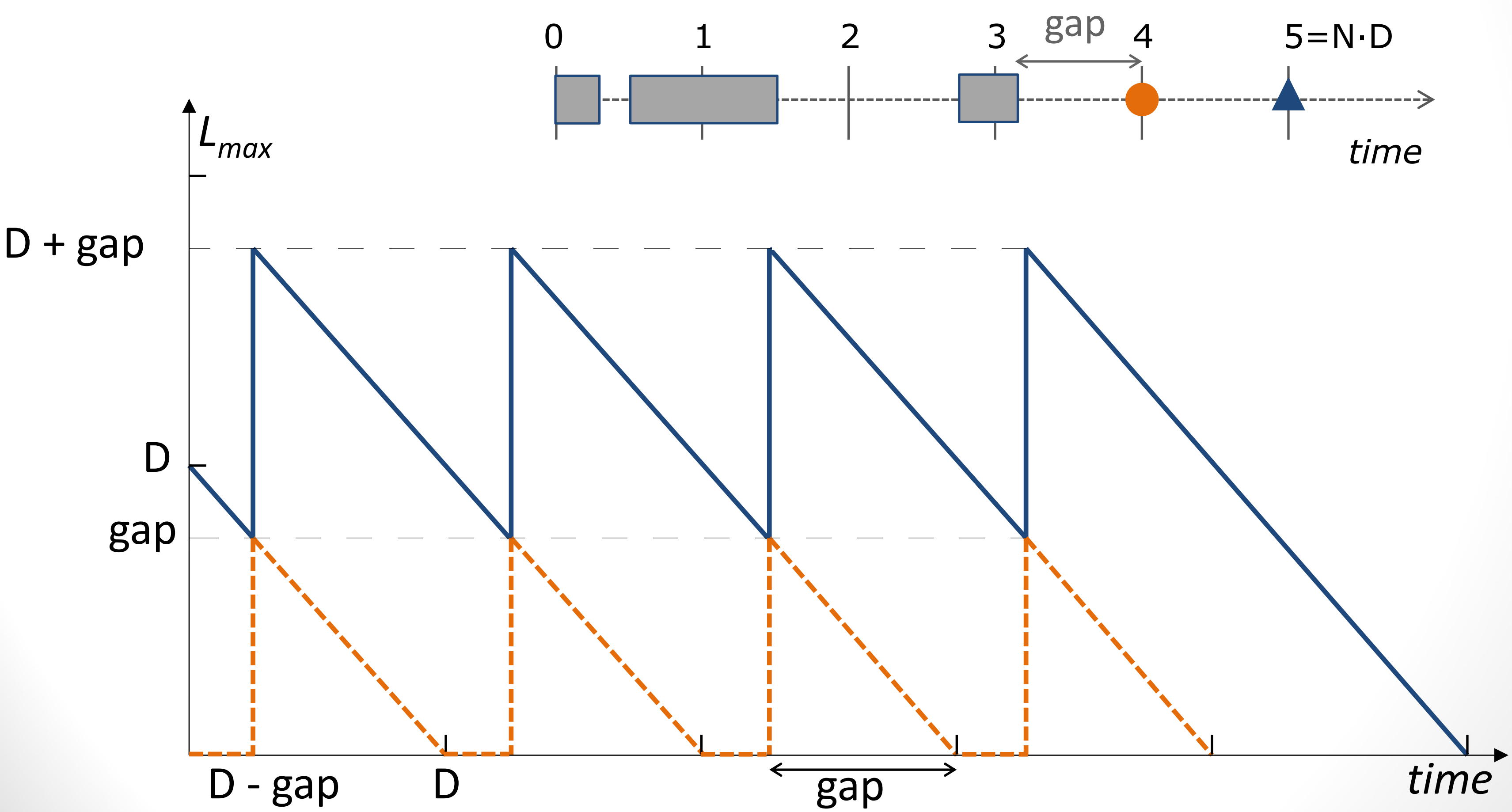
The value (=negative cost) of a void should be related to its future fillability

Depends on:
- size of void
- time until expiration
- packet size distribution

Added value of ▲ over ● depends on same variables

Lifecycle of a void

shows the maximum packet size $L_{max}(t)$ that fits in the void (assuming no prior arrival already filled it)



Theoretical void values

- different packet size distributions possible (E[B]=D)
- exponentially distributed interarrival times (Poisson process E[T]=1/λ)
- load ρ=E[B]/E[T]
- arrivals that fit into void are inhomogeneous Poisson process

  with arrival rate $\lambda(t) = \lambda \cdot Prob[B \leq L_{max}(t)]$
- overall expected number of packets that could fill the void during its life

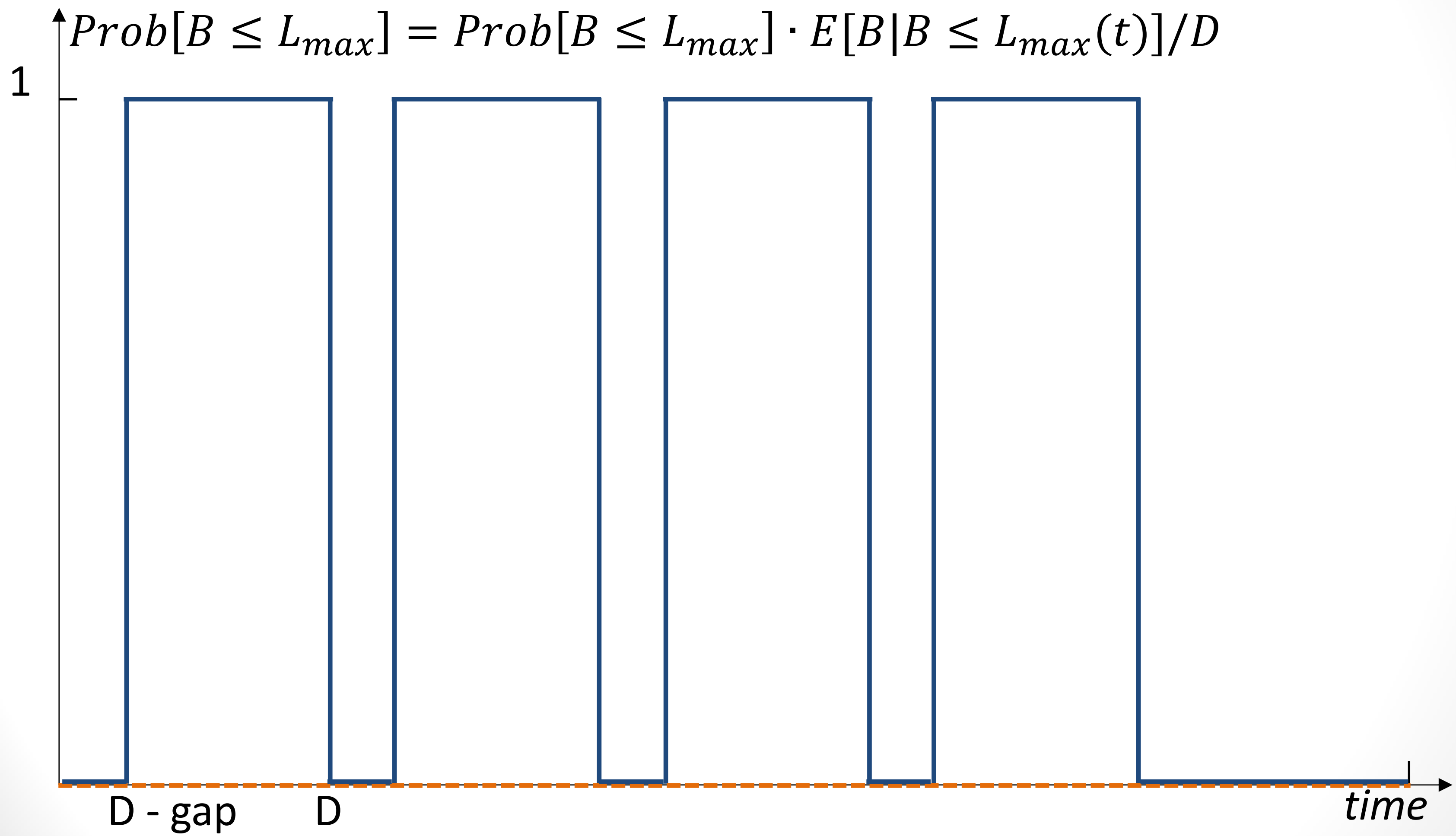
  cycle = integrating λ(t) over the life cycle

$$\Lambda = \int \lambda(t)dt = \frac{\rho}{D} \cdot \int_0^\infty Prob[B \leq L_{max}(t)]dt$$

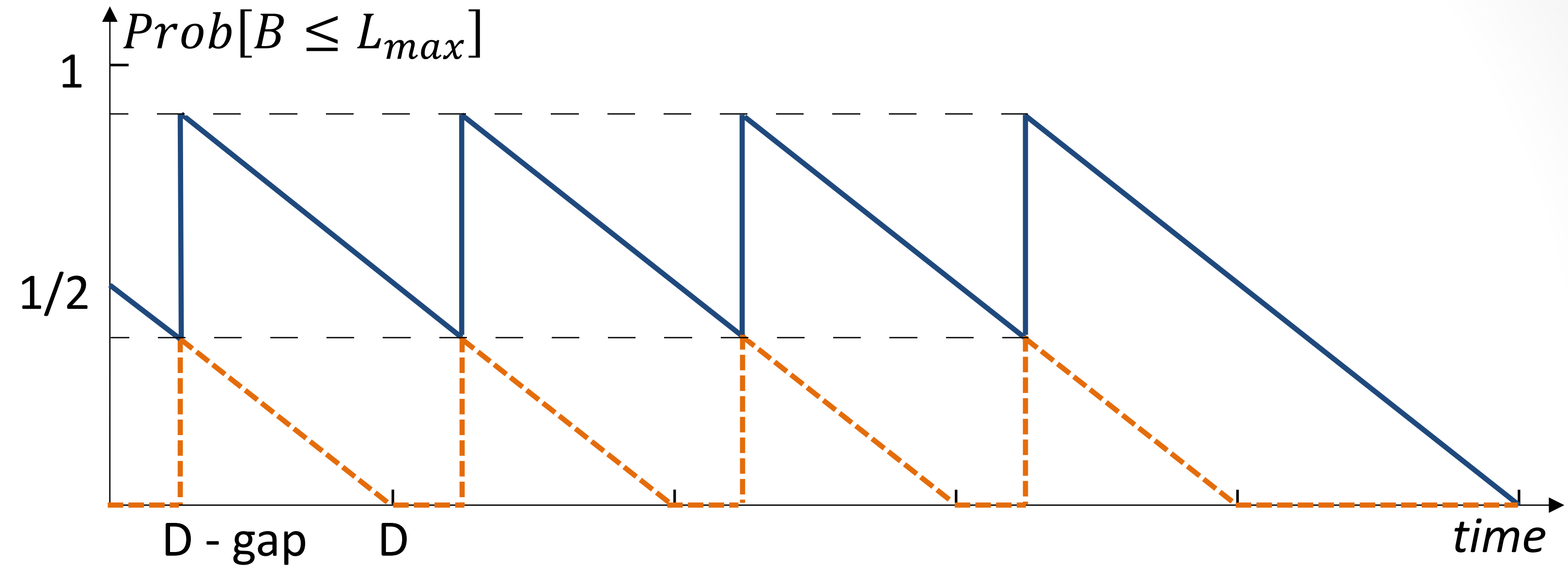
- overall normalized expected packet size of those arrivals:

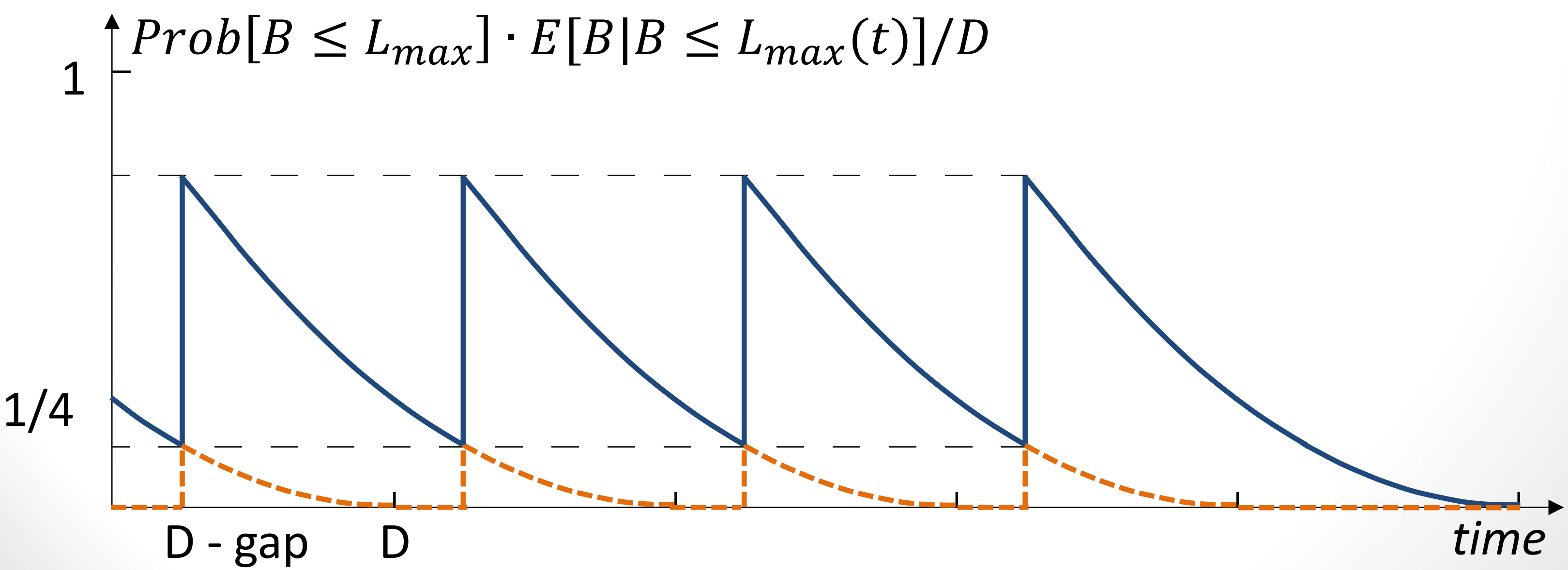
$$\bar{\Lambda} = \int \bar{\lambda}(t)dt = \frac{\rho}{D} \cdot \int_0^\infty Prob[B \leq L_{max}(t)] \cdot \frac{E[B|B \leq L_{max}(t)]}{D} \mathrm{dt}$$

Example: fixed packet size B=E[B]=D

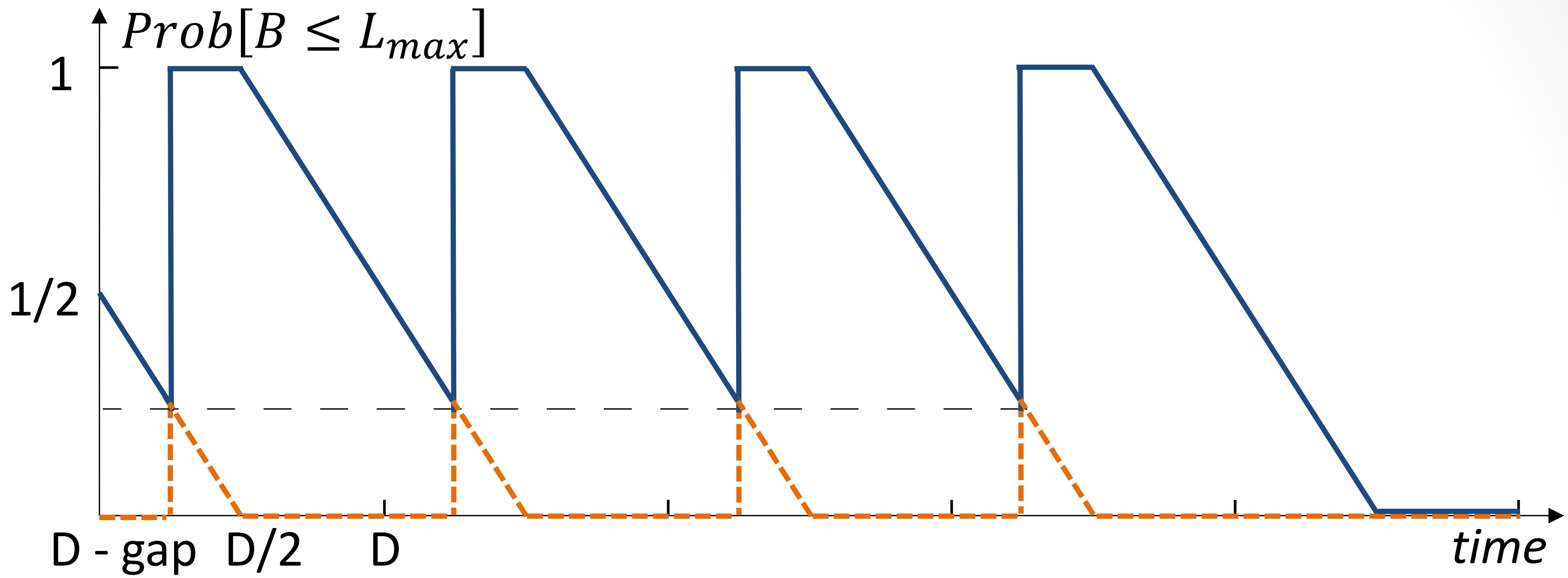


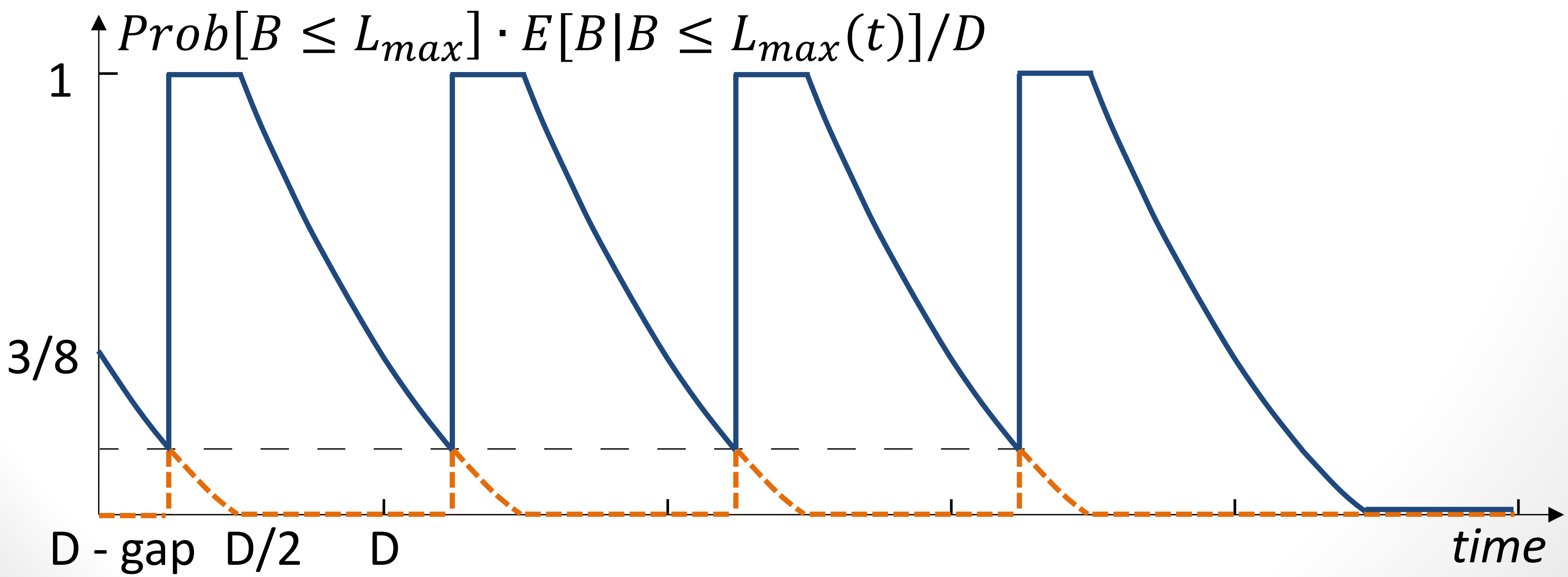
Example: uniform on [0, 2·D]



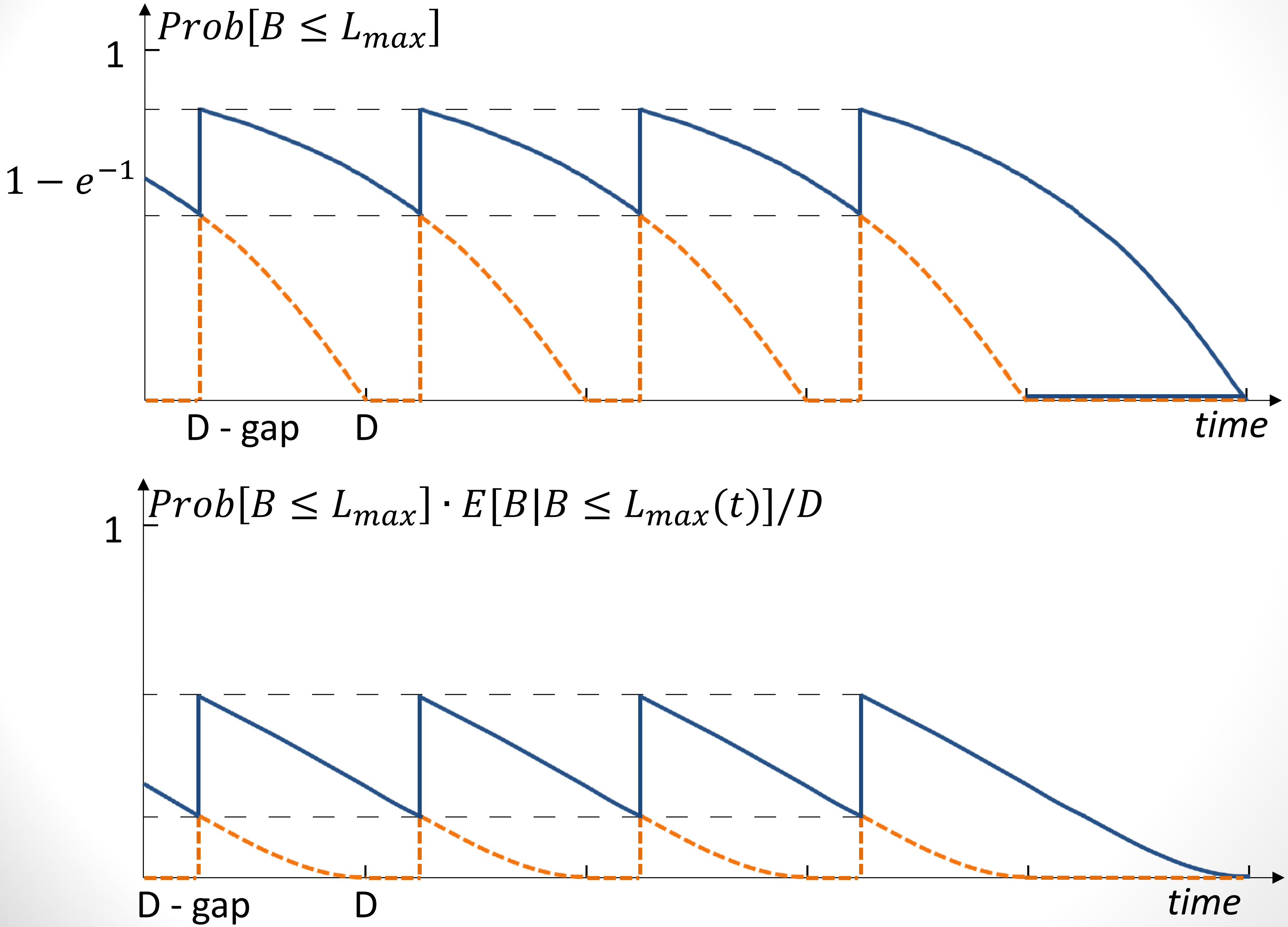


Example: uniform on [0.5 ·D, 1.5·D]

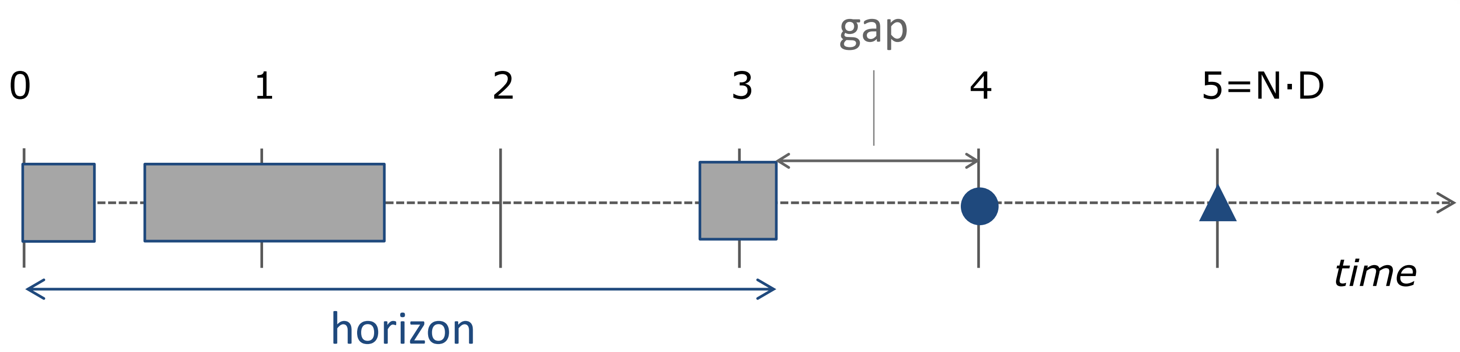




Example: exp. dist. with E[B]=D



Choosing between ▲ and ●



Added value of ▲ over ● = $\overline{\Lambda}$(▲) - $\overline{\Lambda}$(●)

The added value is related to the difference in future fillability

Depends on:
- size of void
- time until expiration
- packet size distribution

Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Assumptions

- **inter-arrival time packets** = exponentially distributed, E[T]
- **packet size distributions**: fixed, uniform [0, 2D] and [0.5D, 1.5D], exp. dist.

  all E[B]=D
- **D** = granularity = 100
- **N+1** = # Fiber Delay Lines = 10
- **load** = $\rho = \frac{E[B]}{E[T]} = 80\ \%$ **AND** 60 %
- **reference algorithm**: schedule on lowest FDL possible (with void filling)
- **void-creating algorithm**:
  - fill void if possible
  - else: schedule on ▲ if $\overline{\Lambda}(▲) - \overline{\Lambda}(●) \geq$ threshold
  - optimize threshold

Performance improvements

maximum gain	*fixed B=D*		*uniform [0, 2D]*		*uniform [0.5D, 1.5D]*		*exp. dist. E[B]=D*	
	ρ=0,6	*ρ=0,8*	*ρ=0,6*	*ρ=0,8*	*ρ=0,6*	*ρ=0,8*	*ρ=0,6*	*ρ=0,8*
LP	-54,1 %	-36,1 %	-5,7 %	-6,9 %	-28,7 %	-19,4 %	-4,5 %	-6,4 %
LPlength	-54,1 %	-36,1 %	-0,6 %	-1,9 %	-25,0 %	-16,5 %	-0,24 %	-0,14 %
delay	-16,3 %	-16,2 %	-1,7 %	-3,9 %	-8,2 %	-8,8 %	-0,8 %	-3,4 %

optimal threshold	*fixed B=D*		*uniform [0, 2D]*		*uniform [0.5D, 1.5D]*		*exp. dist. E[B]=D*	
	ρ=0,6	*ρ=0,8*	*ρ=0,6*	*ρ=0,8*	*ρ=0,6*	*ρ=0,8*	*ρ=0,6*	*ρ=0,8*
LP	1,0	1,2	1,4	1,6	1,2	1,5	1,1	1,4
LPlength	1,0	1,2	1,9	2,0	1,3	1,6	2,3	3,0
delay	1,0	1,2	1,1	1,0	1,2	1,1	0,9	0,9

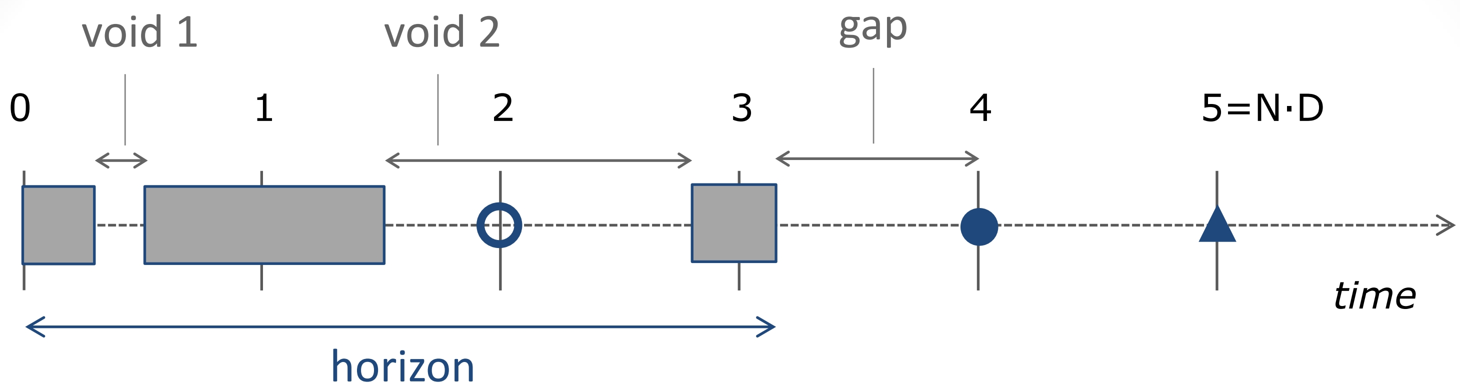
Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

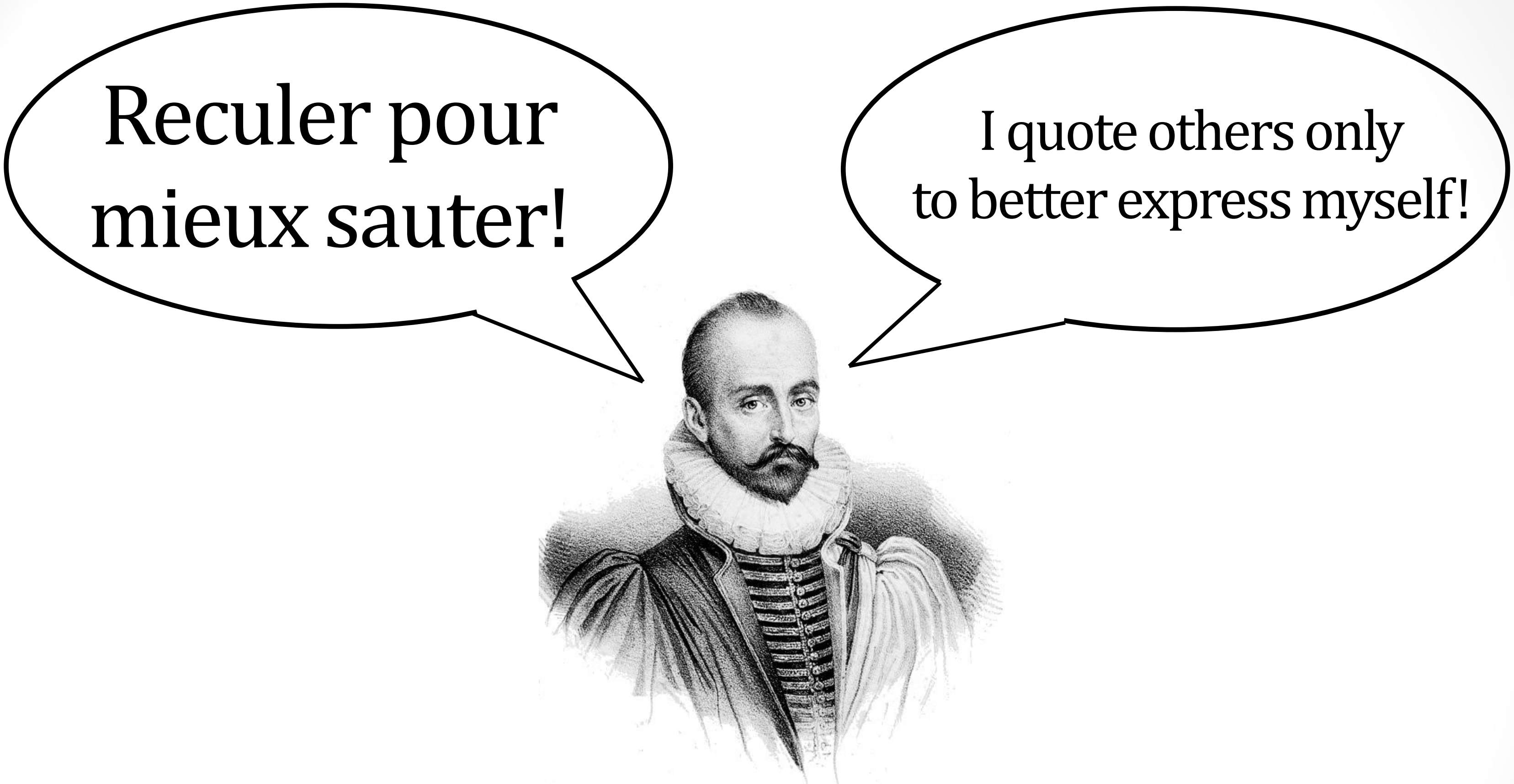
Overview presentation

- Contention resolution & scheduling basics
- Void-creating scheduling algorithm
- Theoretical void values
- Performance results
- Conclusions

Conclusions



- current algorithms: fill void if possible (○), else ●
- **void creating** algorithms: fill void if possible (○) else ● or ▲
- ▲ : creates larger void
- speculating on **more dense stacking** ⇨ loss probability and delay ↘
- choose between ● and ▲ based on added value $\overline{\Lambda}(▲) - \overline{\Lambda}(●) \geq$ threshold
- **performance improvements** up to 54 % LP reduction
- **optimal threshold** depends on packet size distribution, load and performance measure


Reculer pour mieux sauter!
I quote others only to better express myself!

Montaigne

Questions

?