Void-creating algorithms for fixed packet length in OPS/OBS

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)

Overview presentation

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

Overview presentation

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

The optical backbone

demand for bandwidth ↗

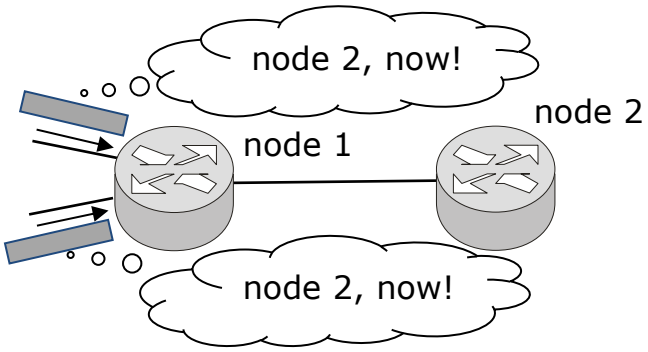
Future: packet-based switching

- shared links
- ✓ improved usage of fiber capacity

- contention possible
  - ✗ potentially, packet loss
  - ✗ no fixed delay
  - ✗ potentially, substantial delay

main motivation of this work:
improving contention resolution

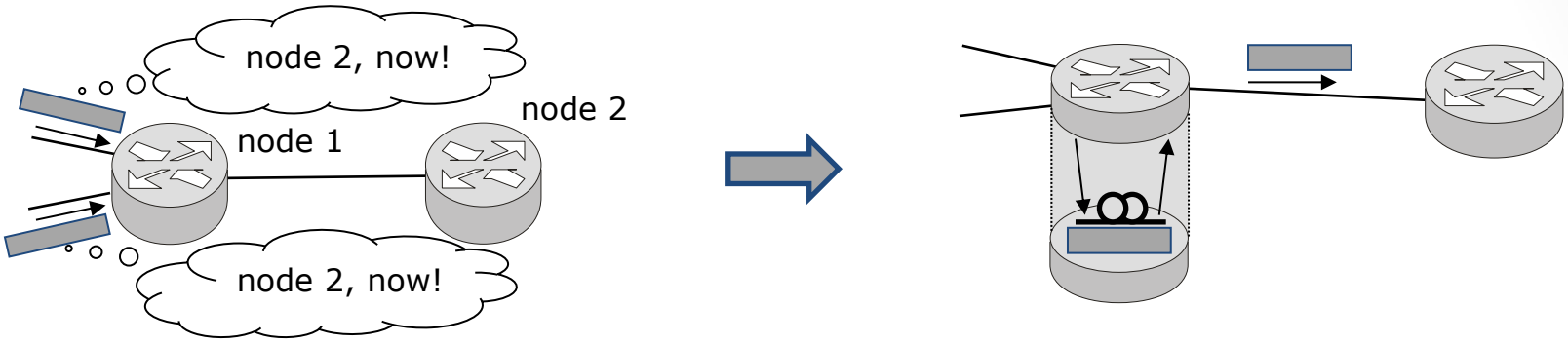
Contention & resolution



Arrival process

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

Contention & resolution



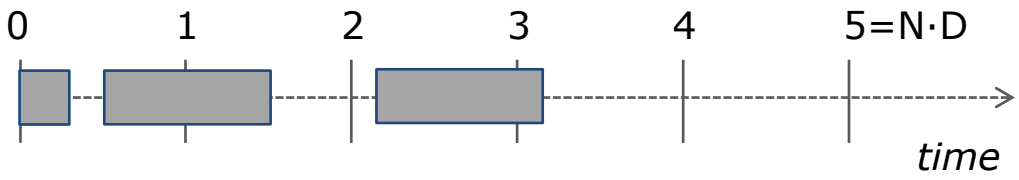
Arrival process

- single wavelength
- fixed 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 = packet lenght = 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

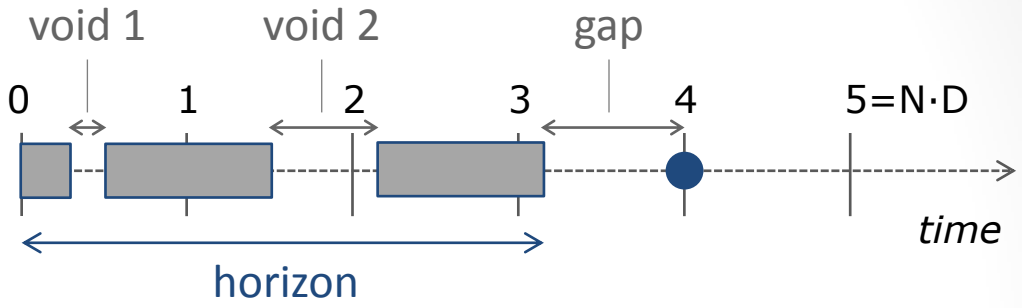
Provisional schedule

choose:

- delay line j (j=0…N)

constraints:

- no overlap



current algorithms: ● : first FDL after horizon

⇩

no fillable voids are created
(=voids larger than packet length B=D)

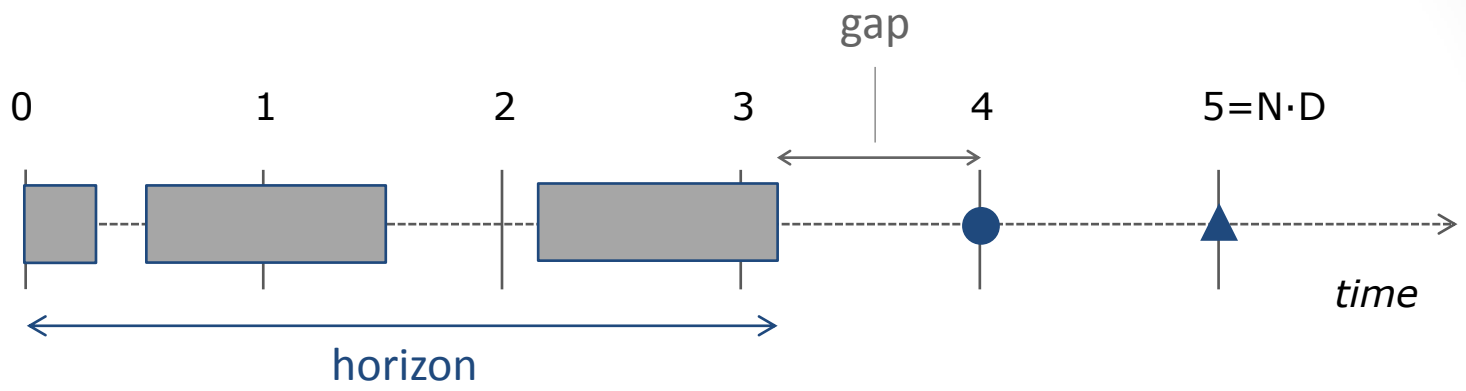
Overview presentation

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

Overview presentation

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

Void-creating scheduling algorithms



choose between

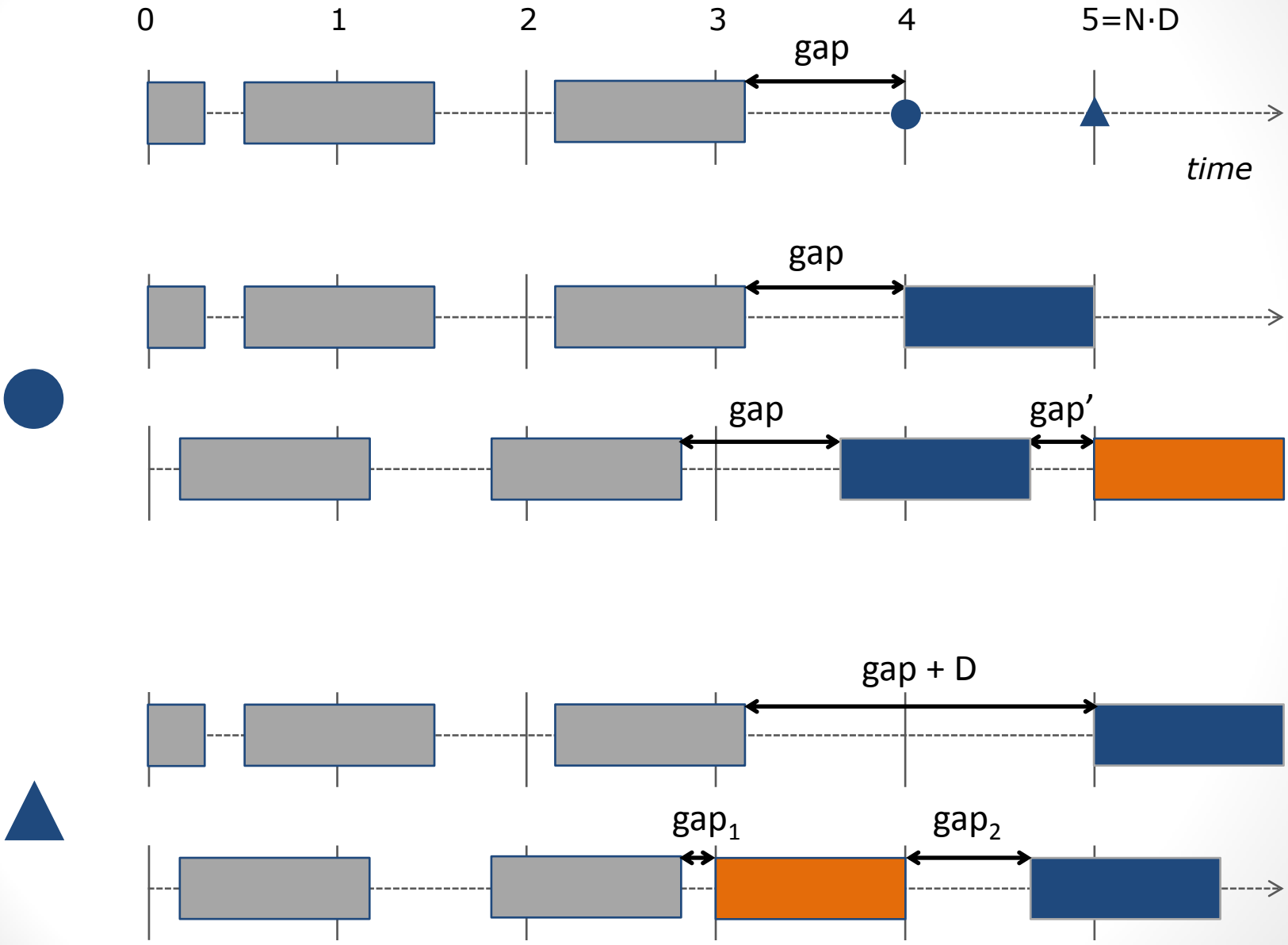
●

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

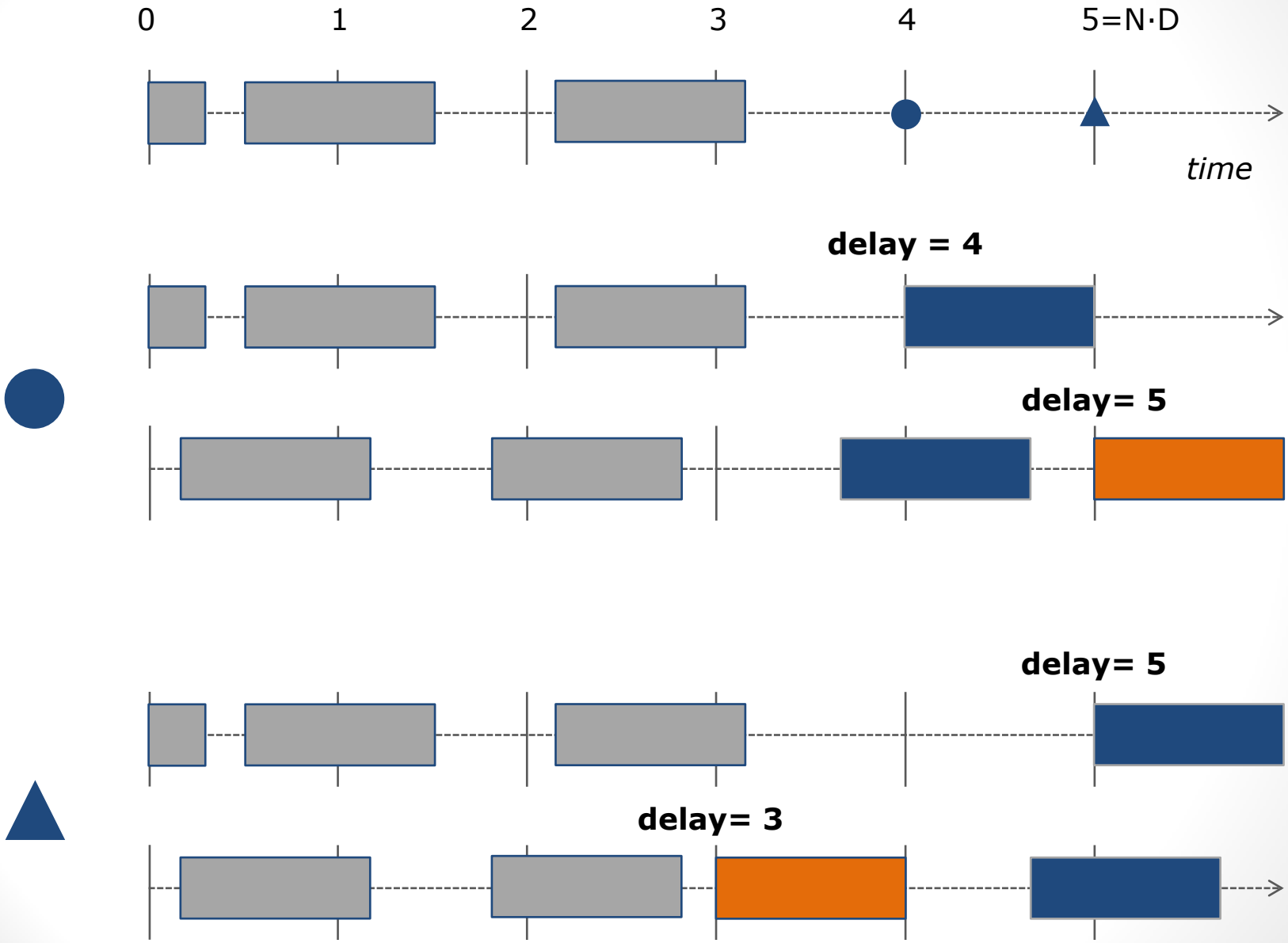
▲

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

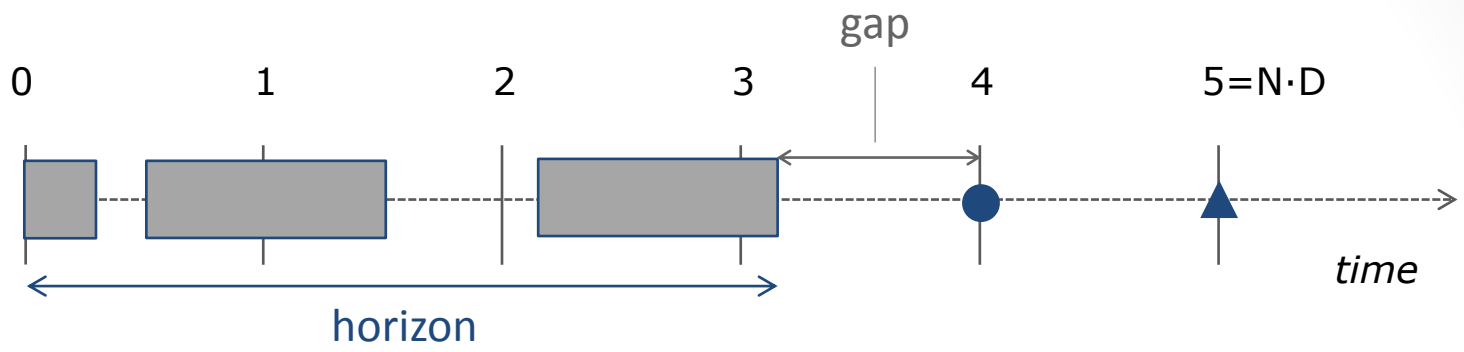
Why ▲ instead of ● ?



Why ▲ instead of ● ?



Why ▲ instead of ● ?



IF fillable void is filled:

- average gap / packet: ↘
- average delay / packet: ↘

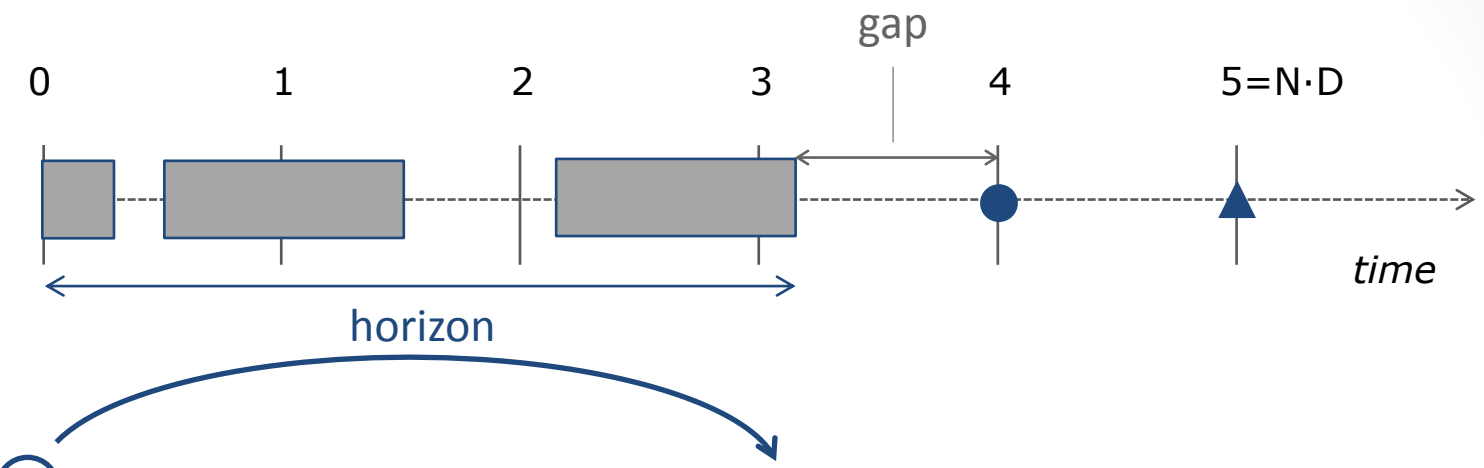
⇩

stacking becomes more dense

⇩

loss probability ↘

Why ▲ instead of ● ?



IF fillable void is filled:

- average gap / packet: ↘
- average delay / packet: ↘

⇩

stacking becomes more dense

⇩

loss probability ↘

not all fillable voids will be filled:

- position with respect to FDL has to be favorable
- depends on arrival instances future packets (stochastic arrival process)
- gap ↗ : chance of filling ↗
  horizon ↗ : chance of filling ↗

Overview presentation

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

Overview presentation

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

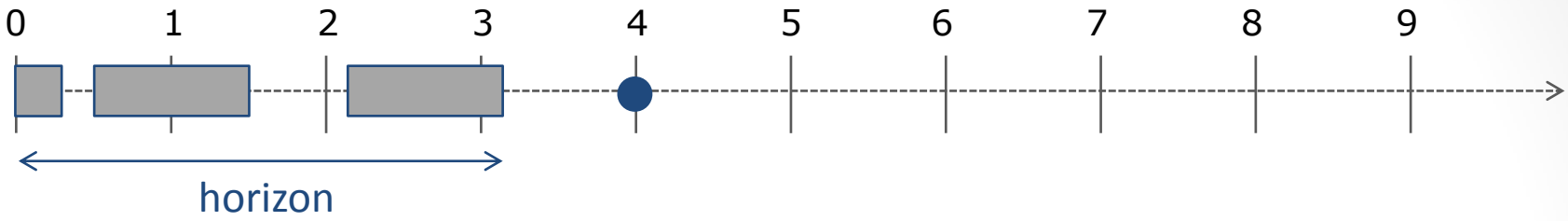
Assumptions

- **inter-arrival time packets** = exponentially distributed, E[T]
- **fixed packet size** = B = 100
- **D** = granularity = 100
- **N+1** = # Fiber Delay Lines = 10
- **load** = $\rho = \frac{B}{E[T]} = 80\ \%$ **AND** 60 %

3 sets of simulations:

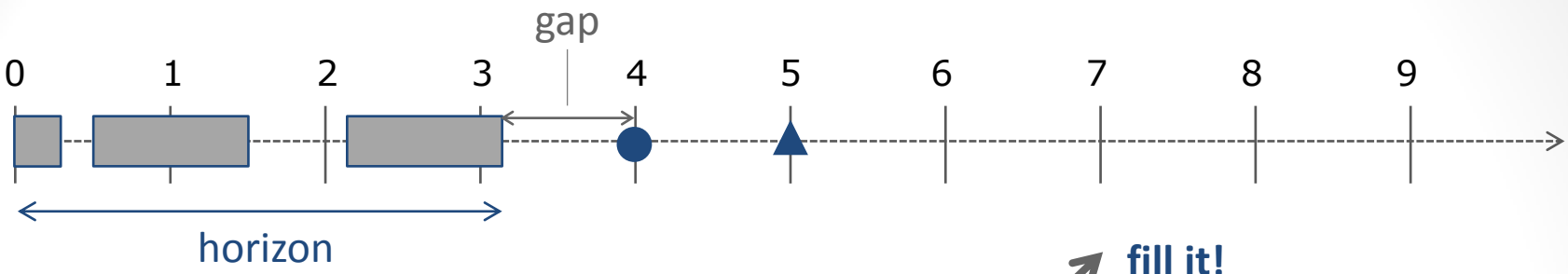
- without void creation
- single creation point
- actual void-creating algorithm

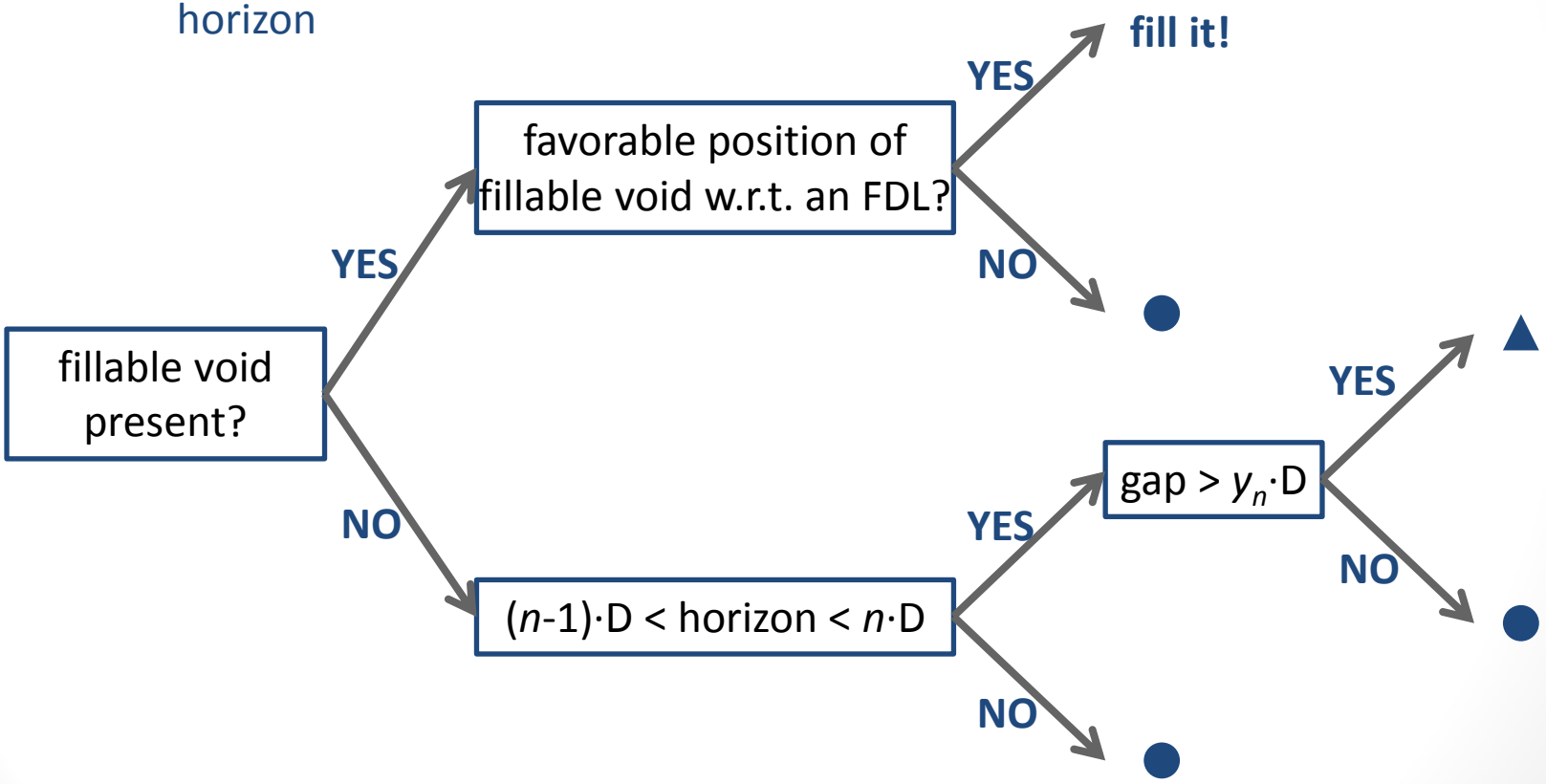
set 1: without void creation



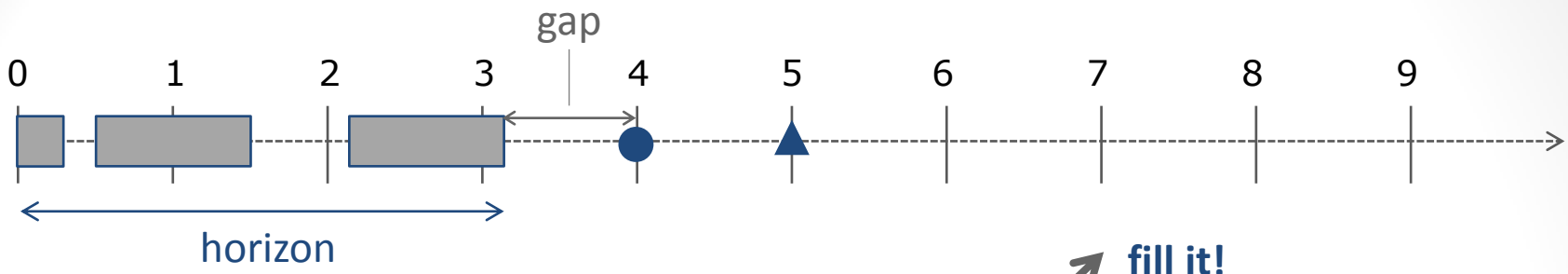
- always ● : first position after horizon
- no fillable voids are created
- $\rho = 80\ \%$: loss probability $= 14{,}46\ \%$
- $\rho = 60\ \%$: loss probability $= 2{,}08\ \%$

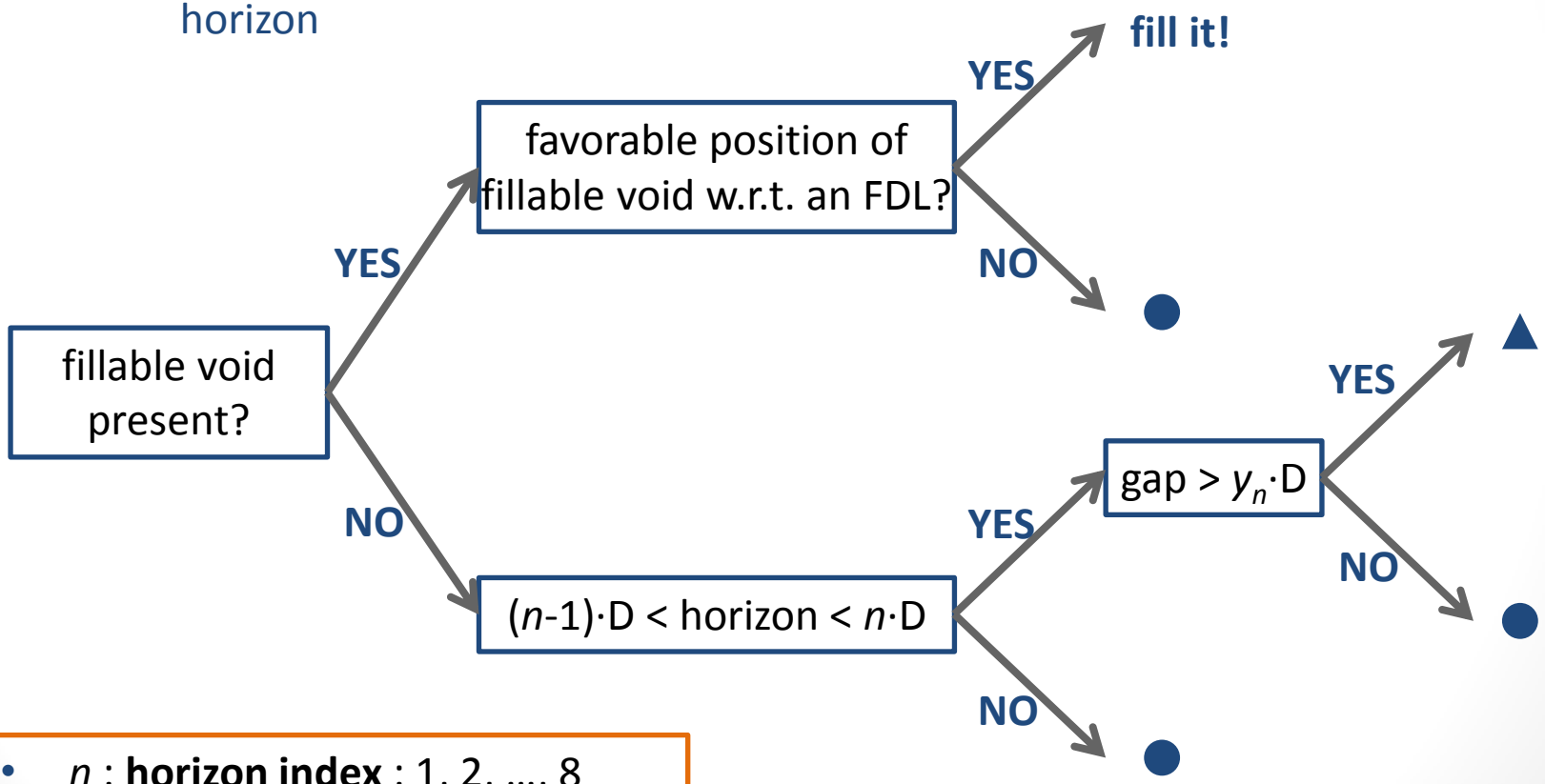
set 2: single creation point





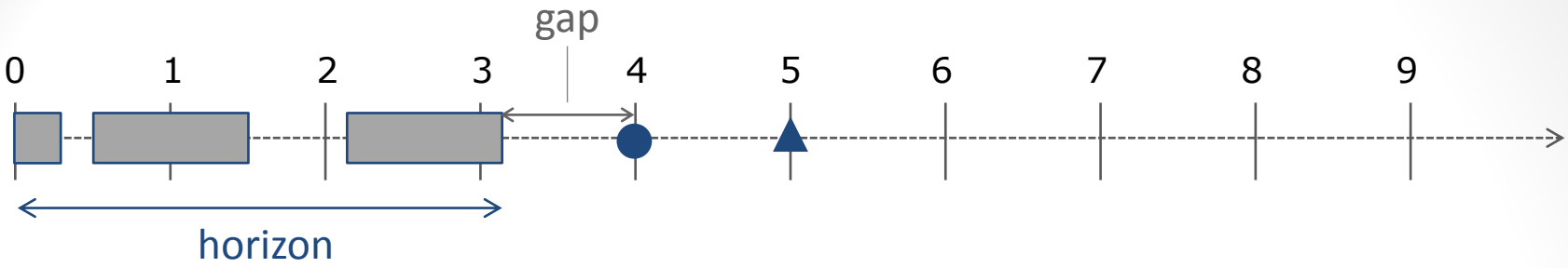
set 2: single creation point



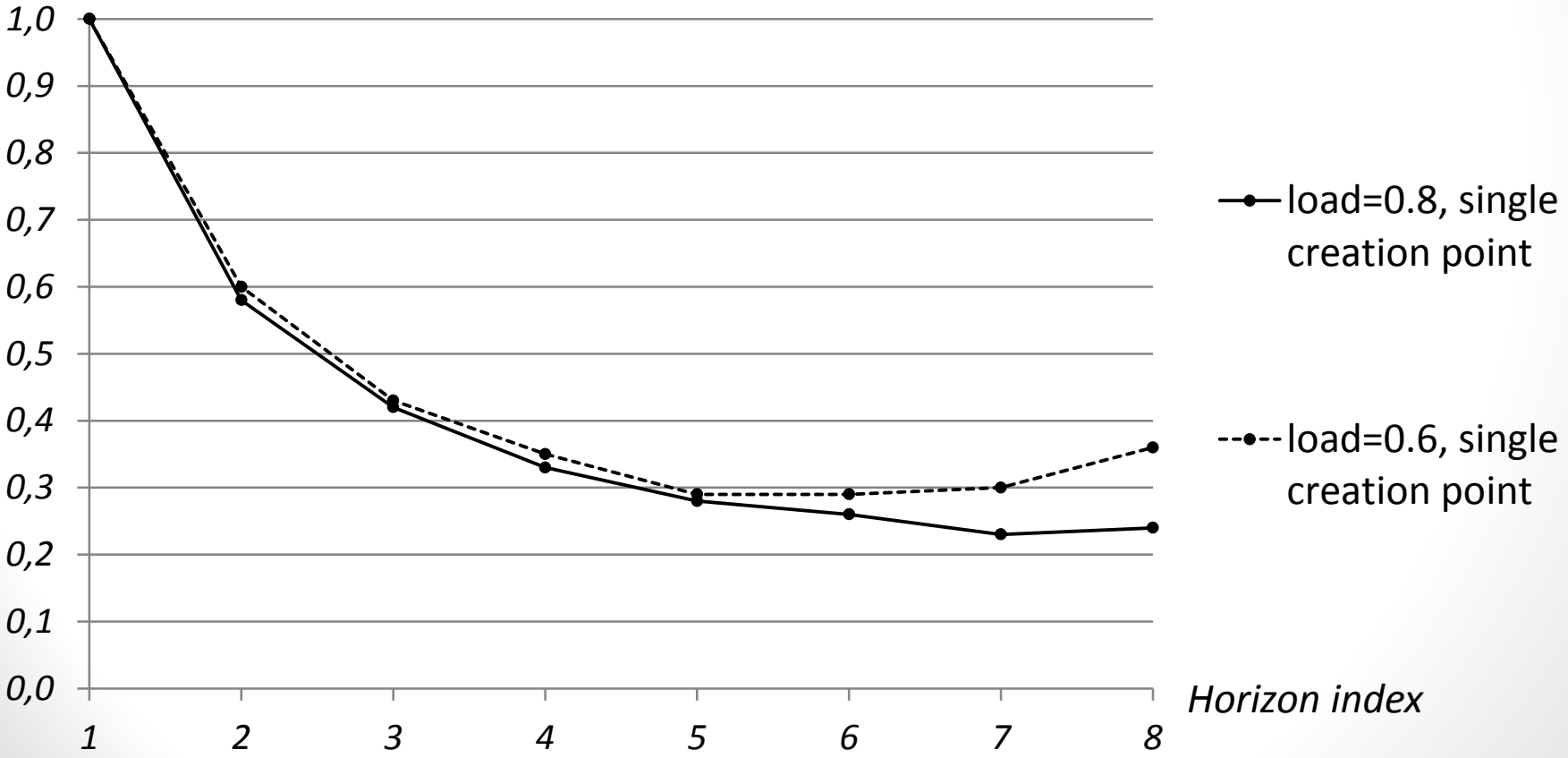


- n : **horizon index** : 1, 2, …, 8
- y_n : **gap threshold** : 0, 0.01, …, 1

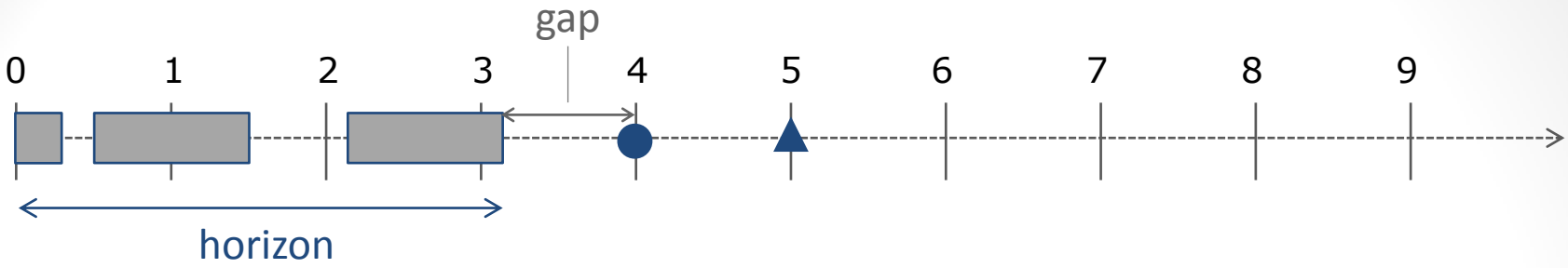
set 2: single creation point

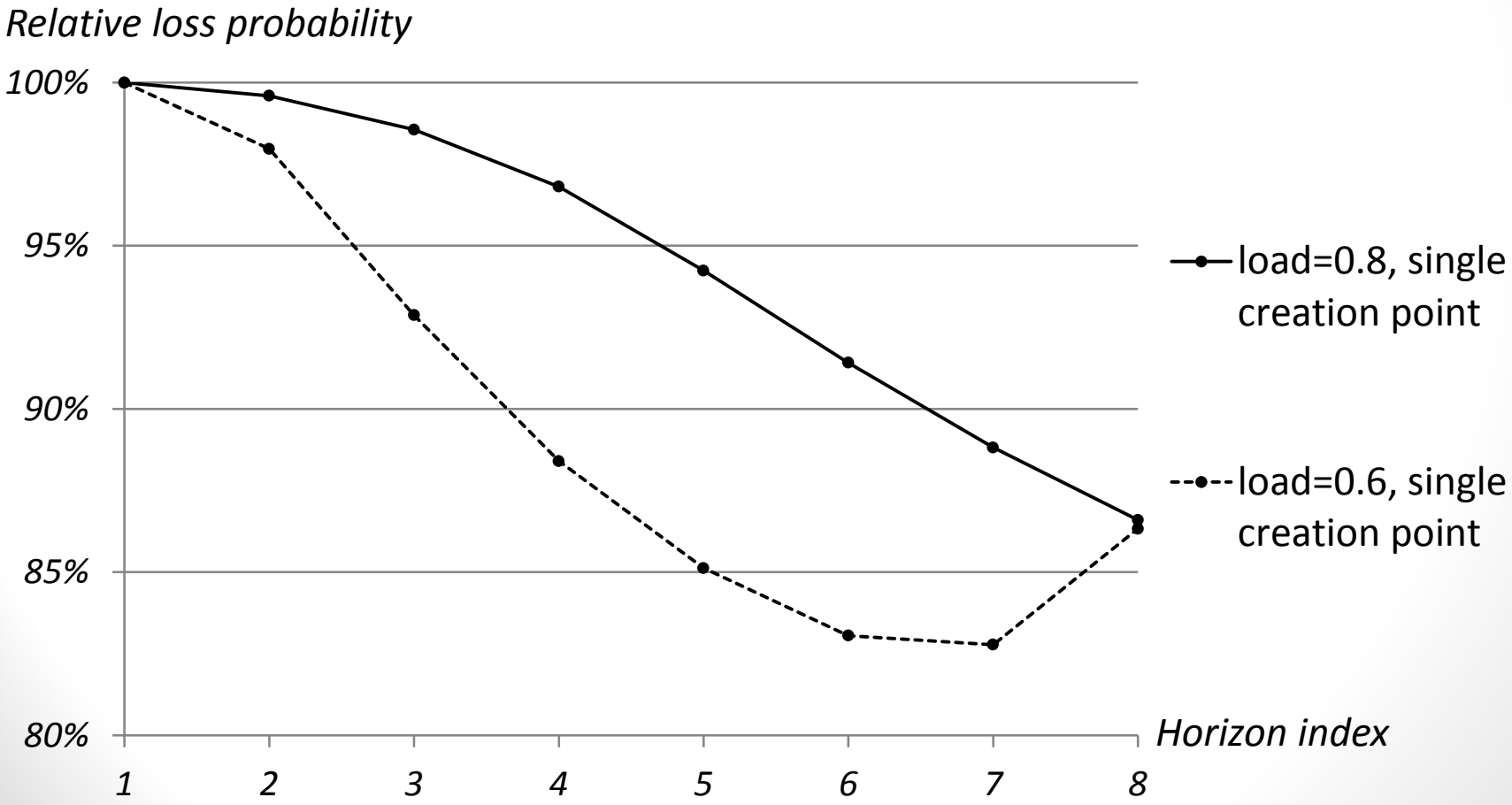


Optimal gap threshold

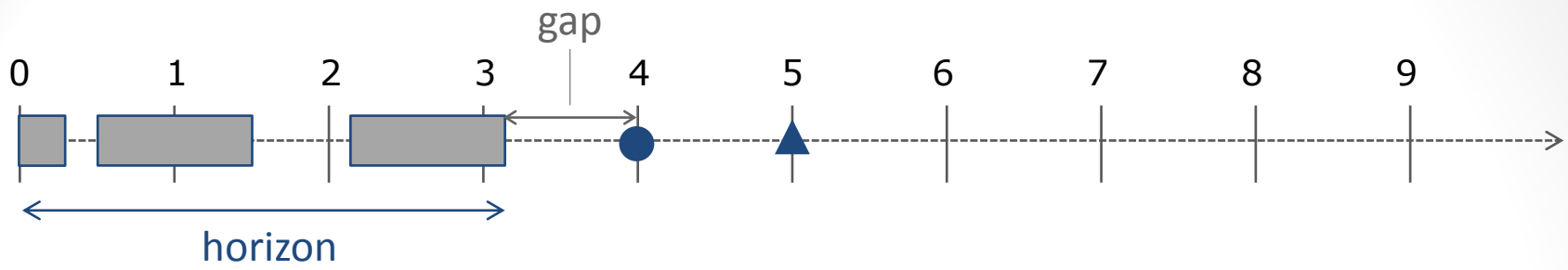


set 2: single creation point

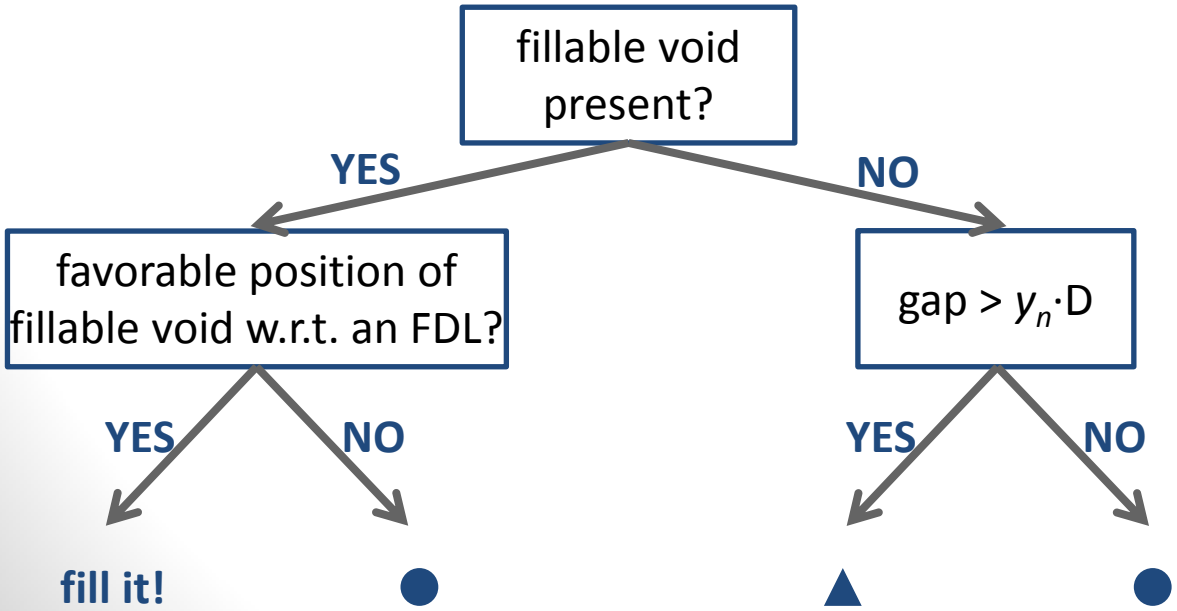




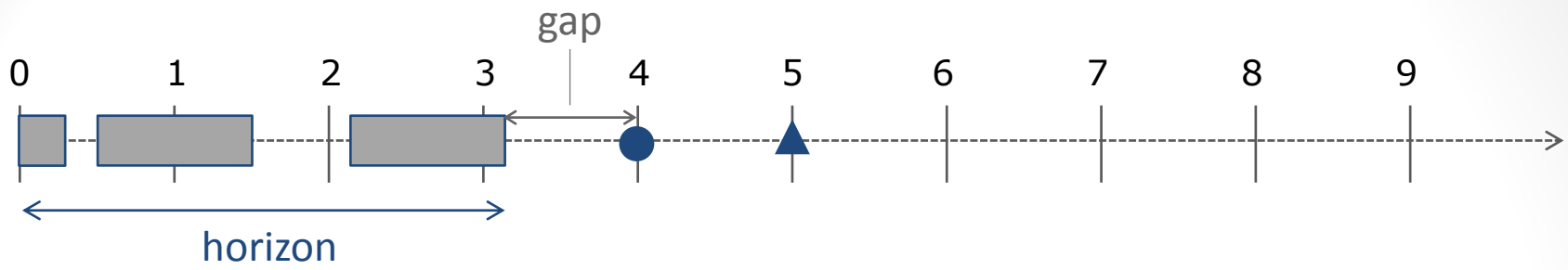
set 3: actual void-creating algorithm



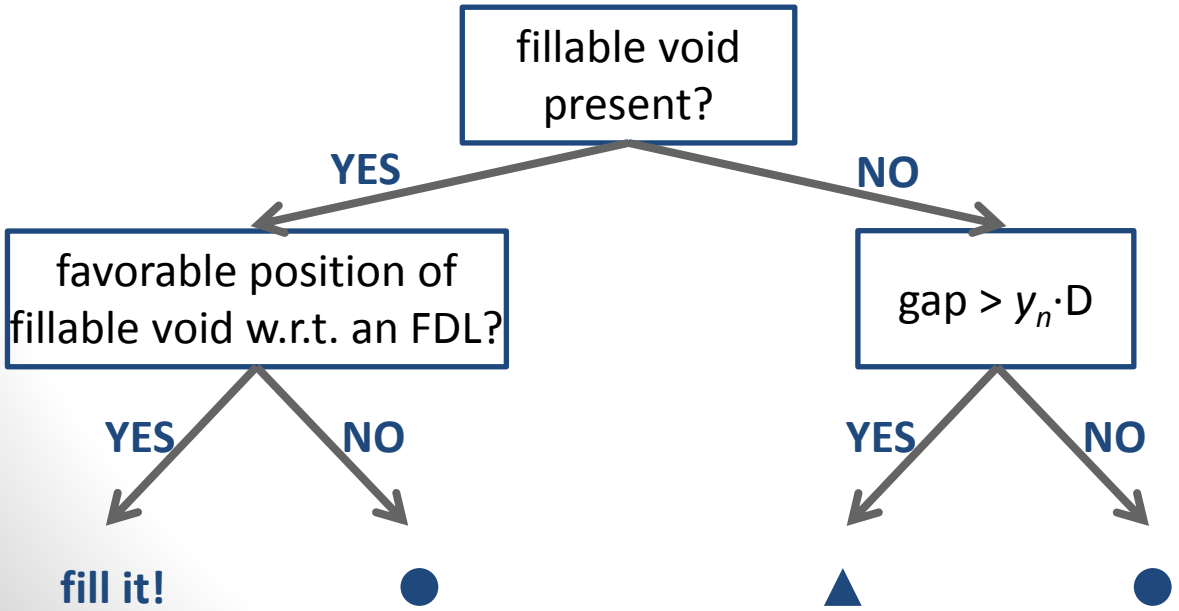
- void creation allowed for all horizon indexes
- optimal gap threshold determined for each horizon index



set 3: actual void-creating algorithm

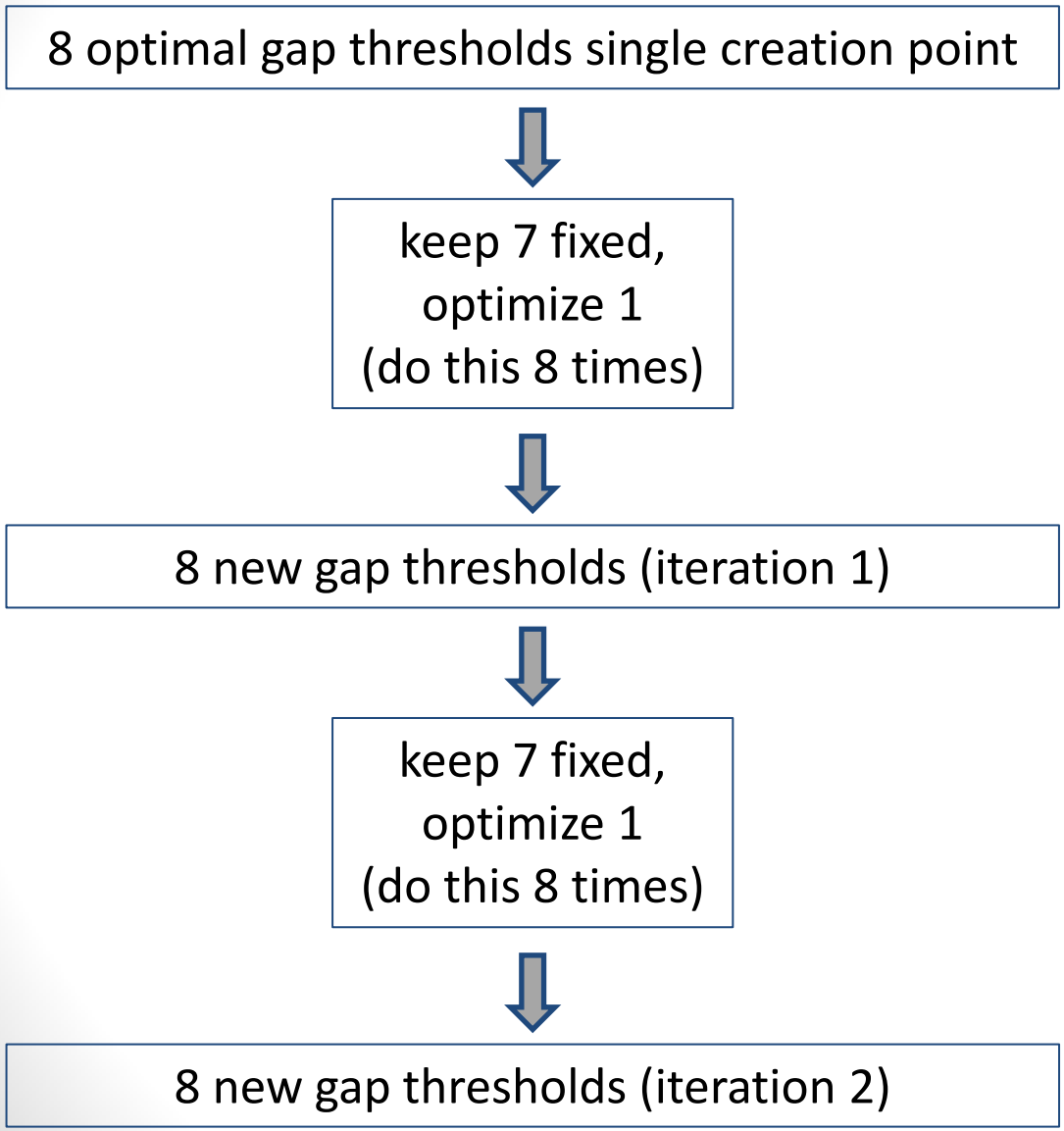


- void creation allowed for all horizon indexes
- optimal gap threshold determined for each horizon index



- y_n : 0, 0.01, …, 1
- parameter space too large (~100^8)

set 3: actual void-creating algorithm



set 3: actual void-creating algorithm

8 optimal gap thresholds single creation point

keep 7 fixed,
optimize 1
(do this 8 times)

8 new gap thresholds (iteration 1)

keep 7 fixed,
optimize 1
(do this 8 times)

8 new gap thresholds (iteration 2)

little difference

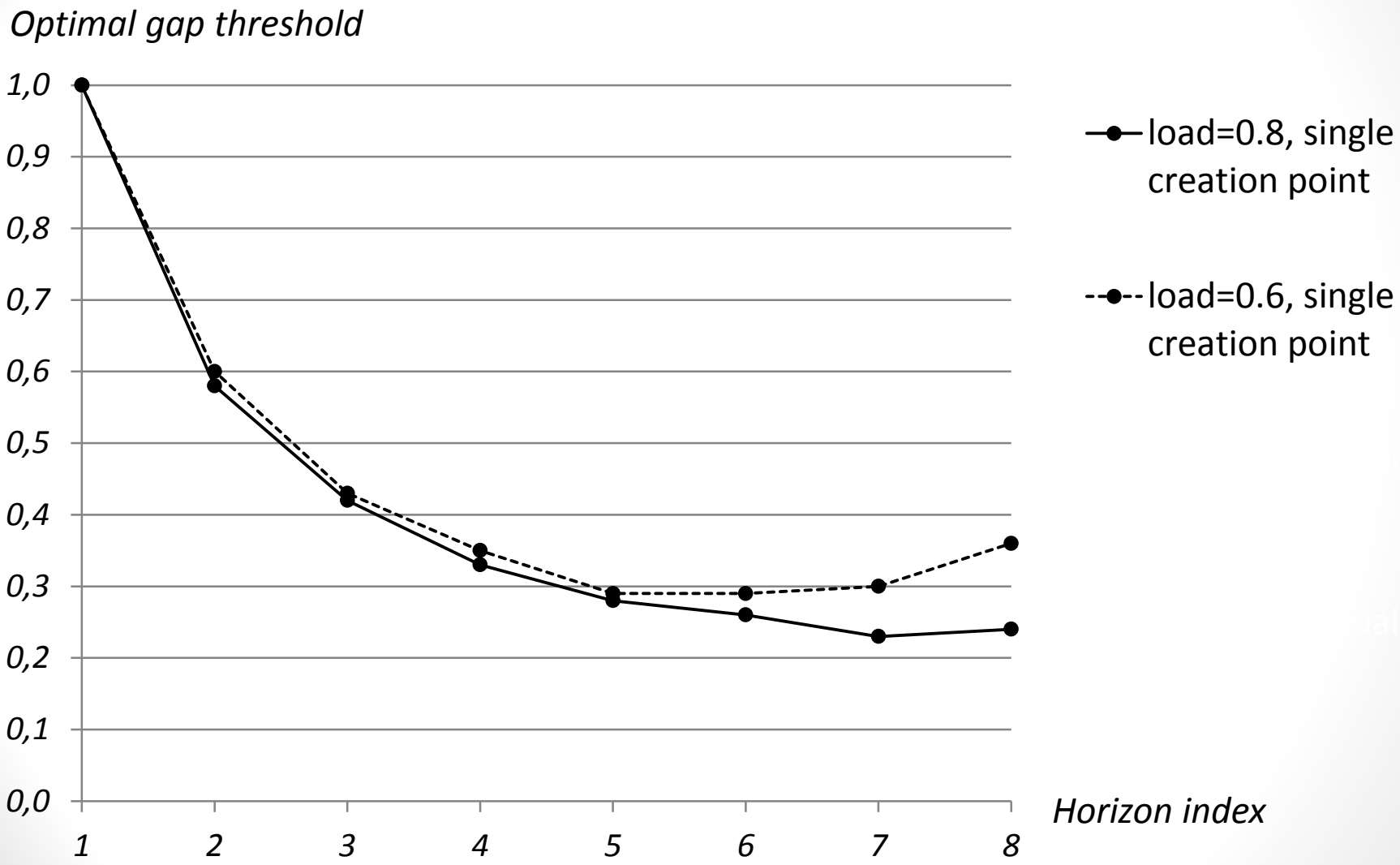
~ convergence

accept iteration 2 as

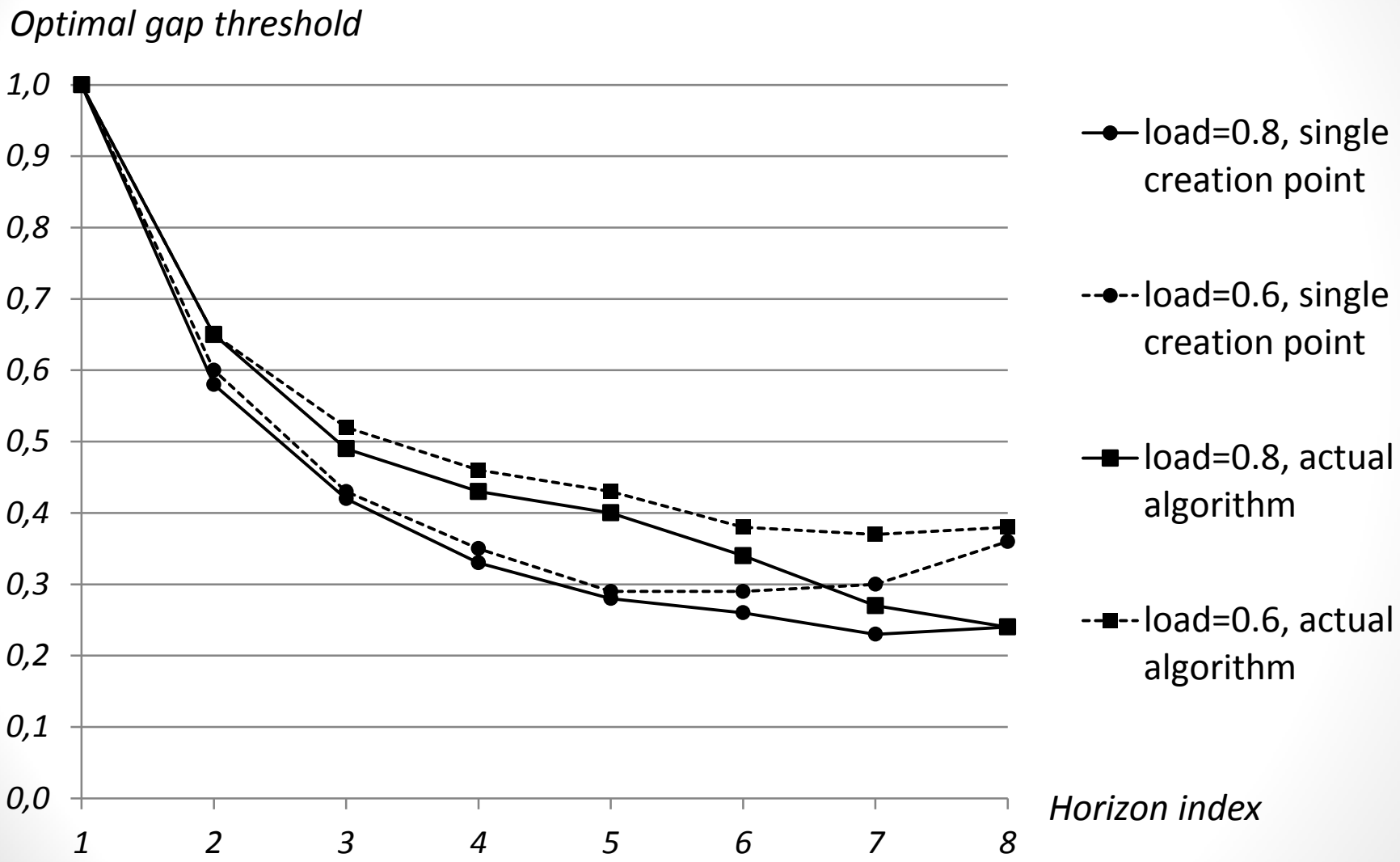
approximation true

optimal gap thresholds

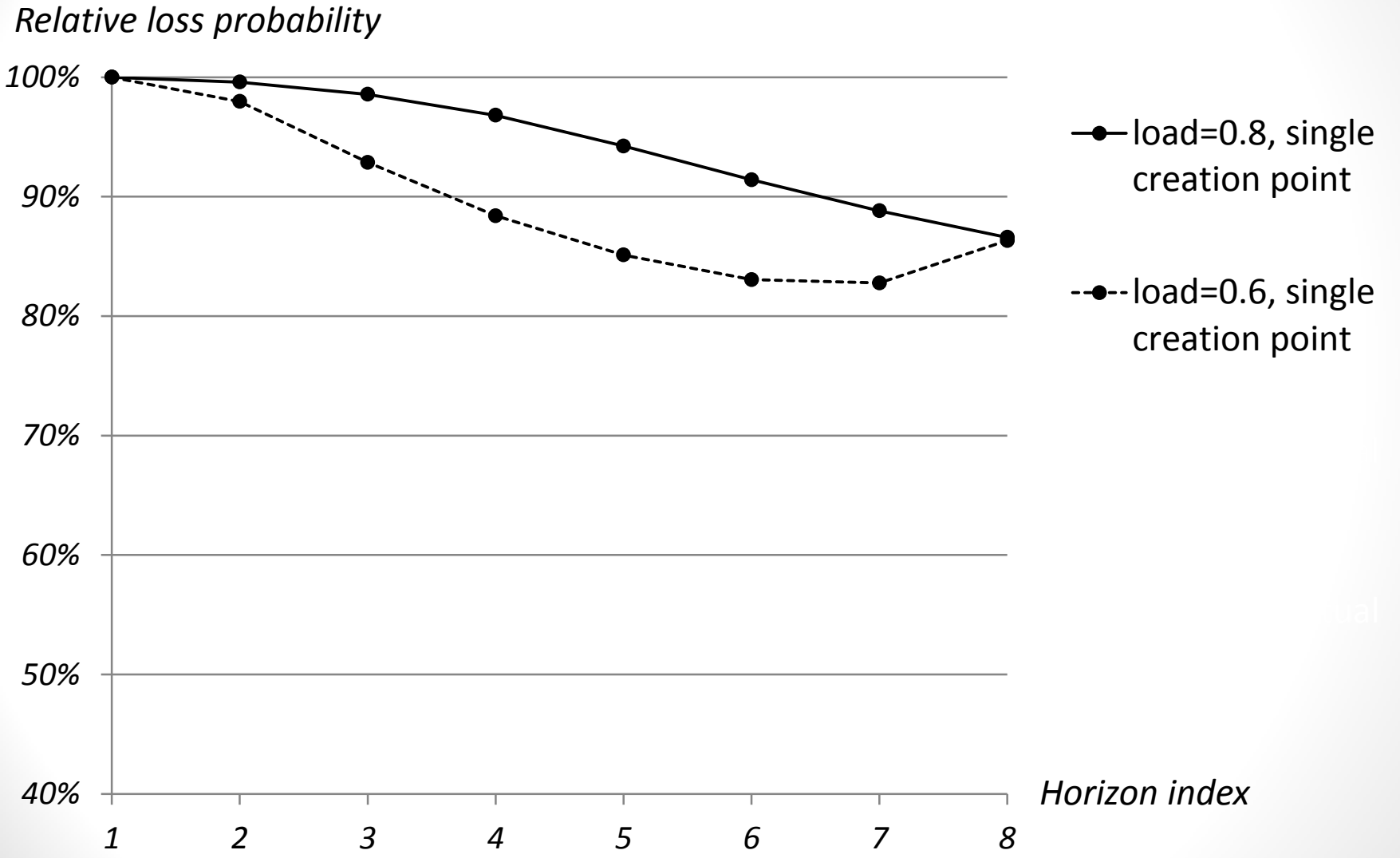
set 3: actual void-creating algorithm



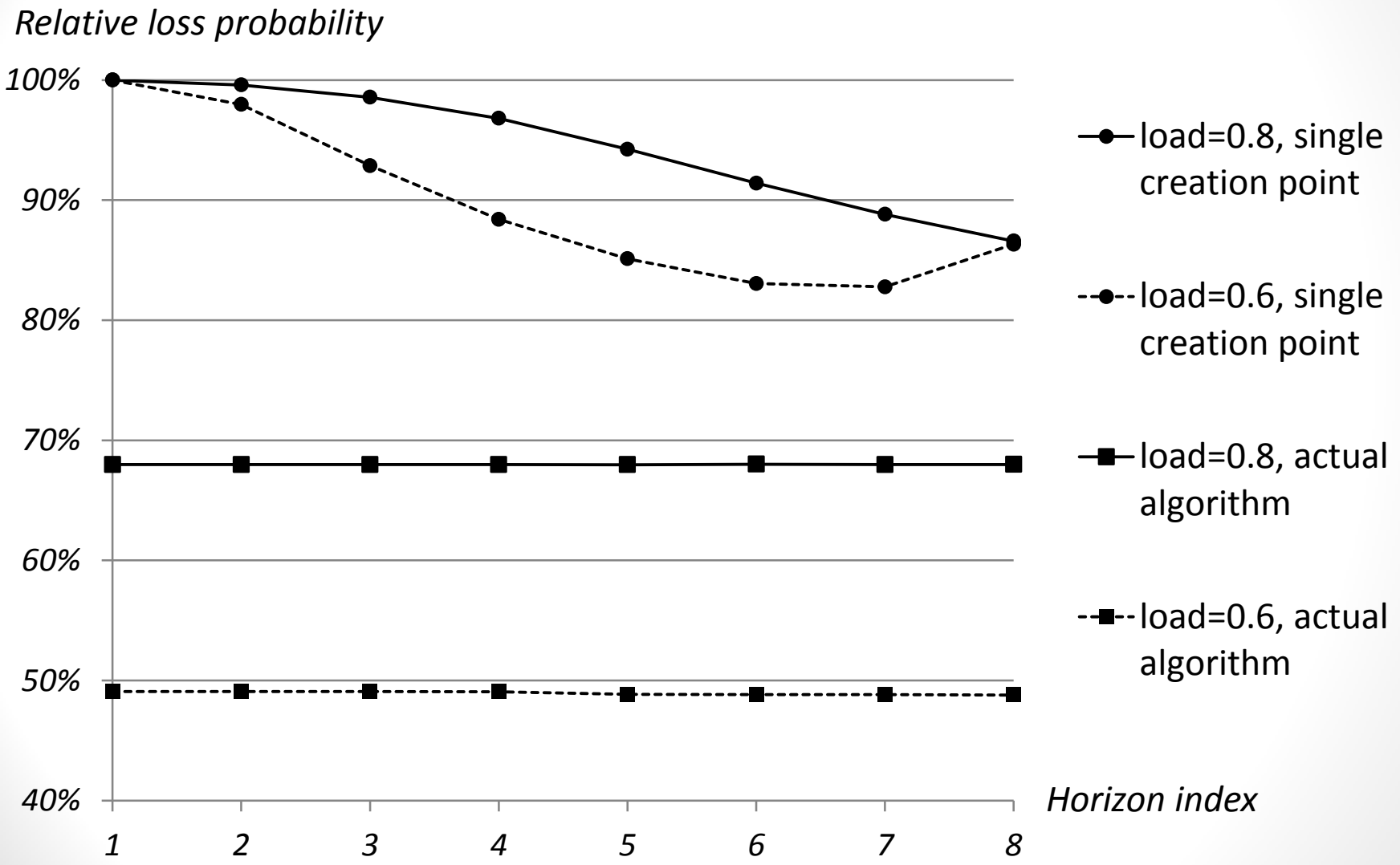
set 3: actual void-creating algorithm



set 3: actual void-creating algorithm



set 3: actual void-creating algorithm



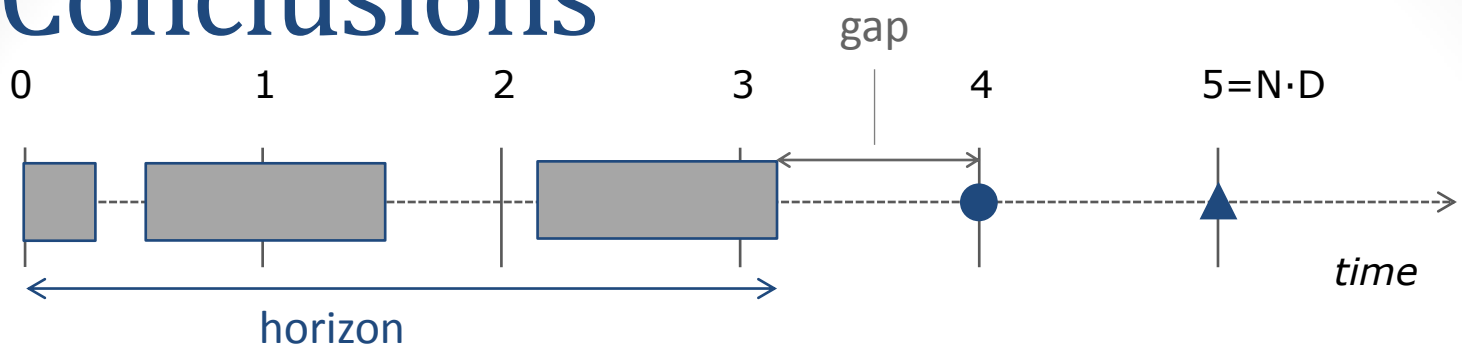
Overview presentation

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

Overview presentation

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

Conclusions



- current algorithms: ●
- void creating algorithms: ● or ▲
- ▲ : creates **fillable void** (1 allowed)
- if filled: more dense stacking ⇨ loss probability ↘
- optimize void creation via **gap threshold** and **horizon index**
- **LP reduction**: load = 80 %: 32 % load = 60 %: 51 %
- **future work**: more complex settings, mathematical approximation

Questions

?