Network Calculus with Compact Domains

Kai Lampka, lampka@it.uu.se

Embedded Systems Group,

Department for Information Technology

Uppsala University, Sweden

Steffen Bondorf, bondorf@cs.uni-kl.de

Jens Schmitt, jschmitt@cs.uni-kl.de

Distributed Computer Systems (DISCO) Lab

Computer Science Department

University of Kaiserslautern, Germany

UPPSALA UNIVERSITET

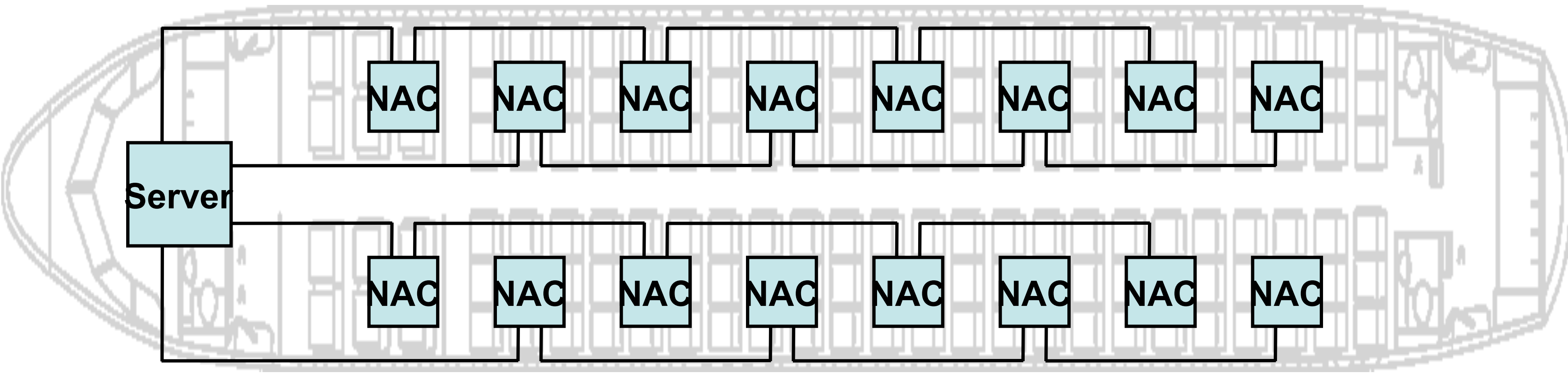
Informationsteknologi

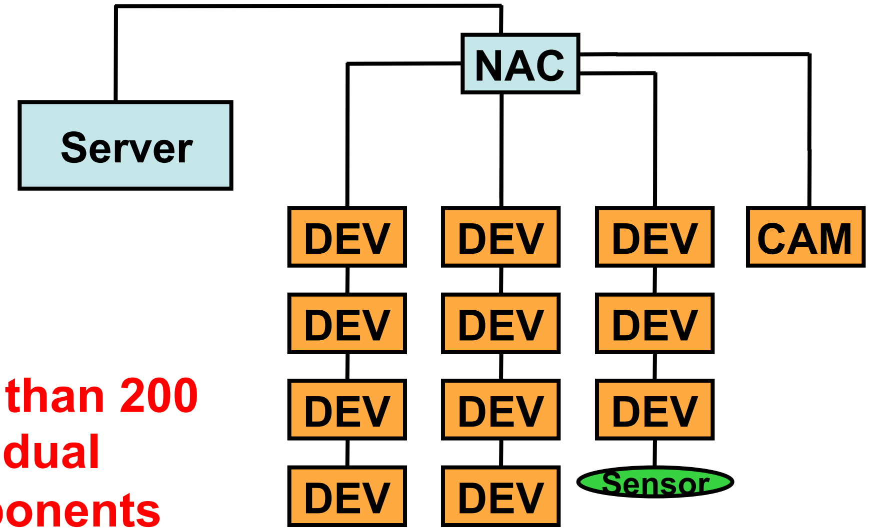
UPPSALA UNIVERSITET

Informationsteknologi

In EU FP7 project far far away, the young modelers had to fight a very large system reluctant to any precise analysis.

Student project at ETHZ 2010: Heterogeneous Communication System (HCS)





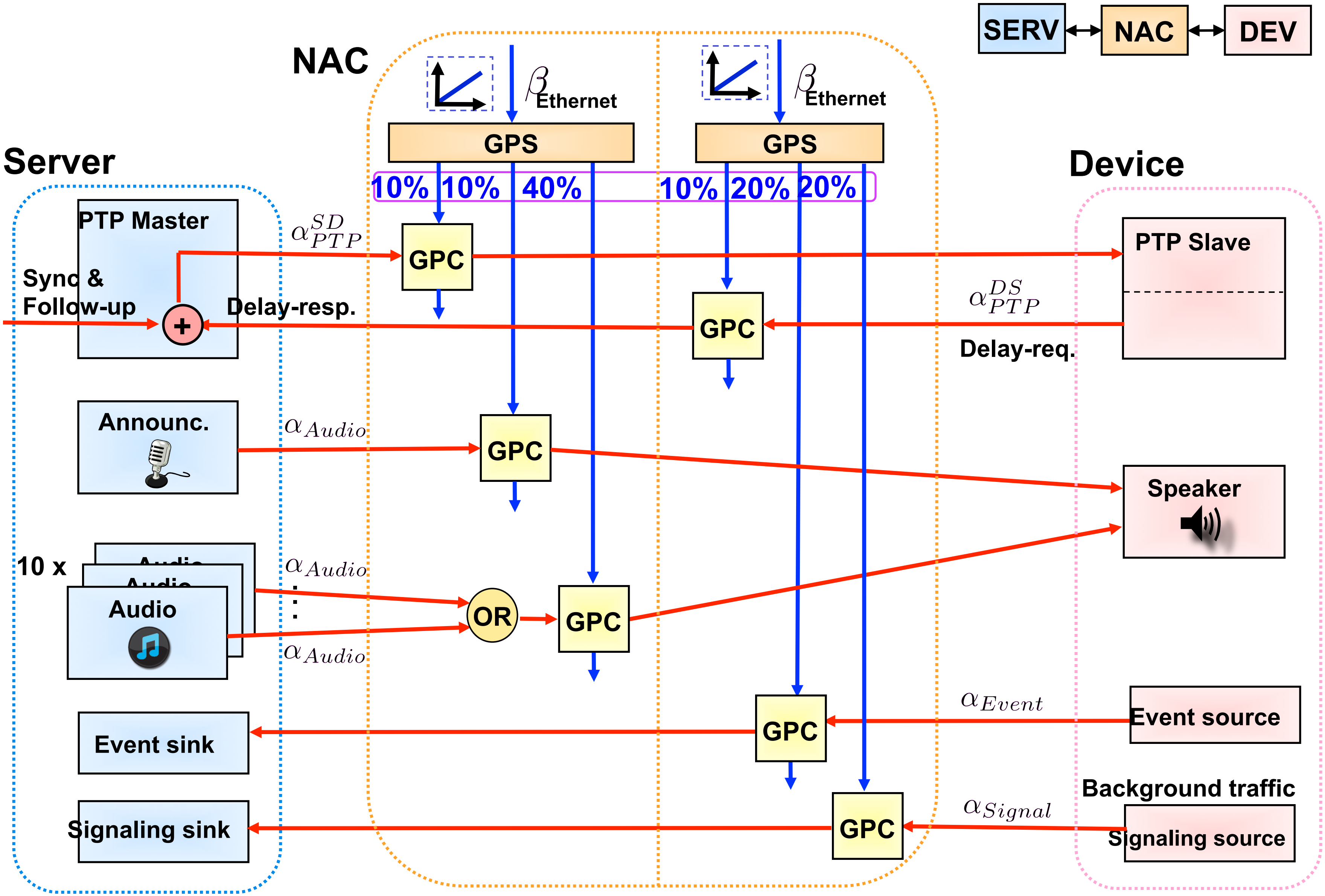
More than 200 individual components

Network traffic:

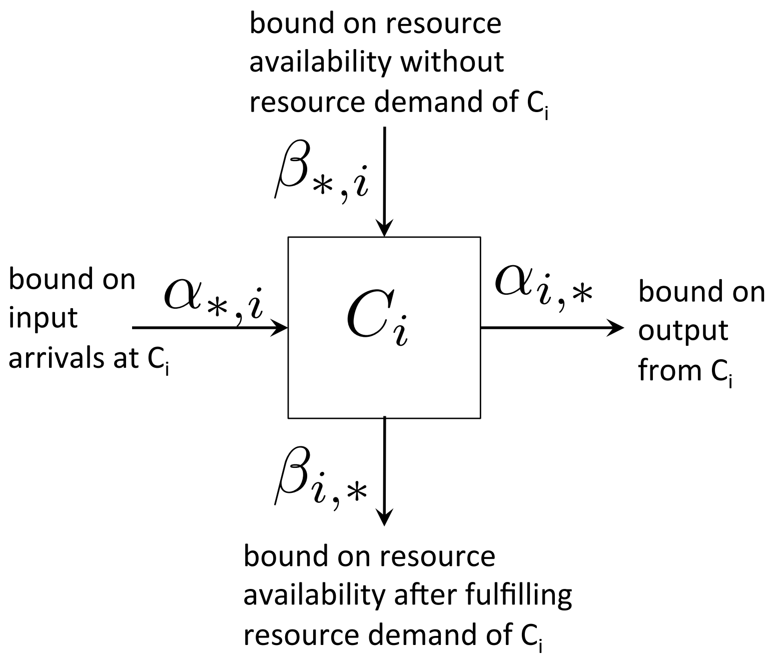
- **Clock synchronization**
- **Audio streaming**
- **Event-based traffic (reading light, …)**
- **Background traffic (network signalling)**

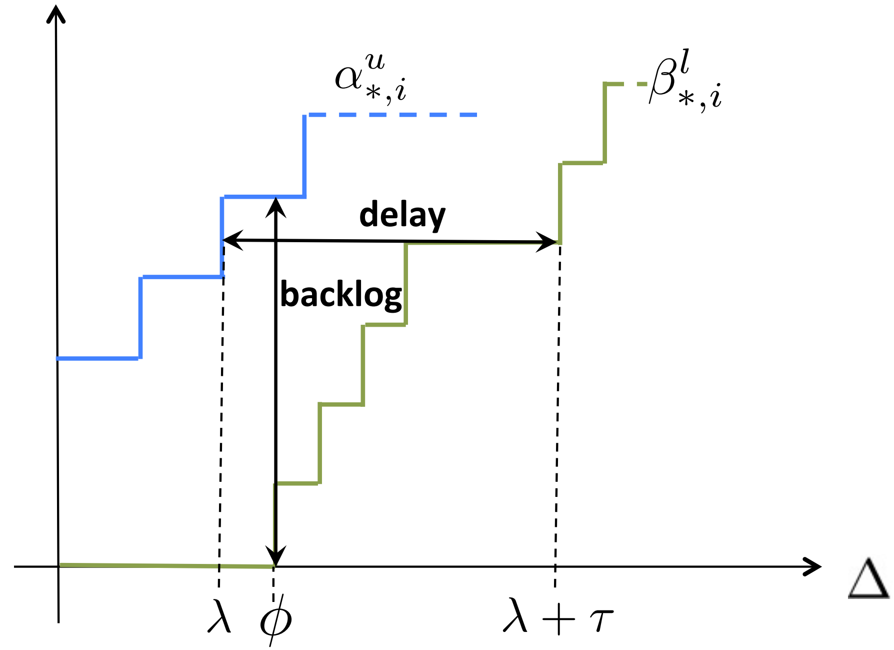
EADS COMBEST

Small scale RTC Model (3 flows-of-intrests)



Greedy Processing Component (GPC),





$$backlog_i \leq \sup_{\phi \geq 0} \{\alpha^u_{*,i}(\phi) - \beta^l_{*,i}(\phi)\}$$

$$delay_i \leq \sup_{\lambda \geq 0} \{\inf\{\tau \geq 0 \; : \; \alpha^u_{*,i}(\lambda) \leq \beta^l_{*,i}(\lambda + \tau)\}\}$$

Flow equations of GPC, [Wandeler'06]

$$\alpha^u_{i,*} = \min\{(\alpha^u_{*,i} \otimes \beta^u_{*,i}) \oslash \beta^l_{*,i}, \beta^u_{i,*}\}$$
$$\alpha^l_{i,*} = \min\{(\alpha^l_{*,i} \oslash \beta^u_{*,i}) \otimes \beta^l_{*,i}, \beta^l_{*,i}\}$$
$$\beta^u_{i,*} = (\beta^u_{*,i} - \alpha^l_{*,i}) \,\overline{\oslash}\, 0$$
$$\beta^l_{i,*} = (\beta^l_{*,i} - \alpha^u_{*,i}) \,\overline{\otimes}\, 0$$

GPC analysis follows a Total Flow Analysis: end-to-end delay computed from sum of GPC-delays

Problems

1) Enormous effort for manually creating an MPA model for the HCS due to its sheer size (number of GPCs)

2) For large systems, the representation of the arrival and services curves gets complex very fast, leading to:

   - Long execution times

   - Large memory consumption ("out of memory"-errors)

Even with up-scaling of curves and simplifications, the HCS model as a set of standard GPCs could not be analyzed

Envisioned Solution

- Automatic generation of the MPA model for the HCS system

- Safe "approximation" of arrival/service curves with simpler curves

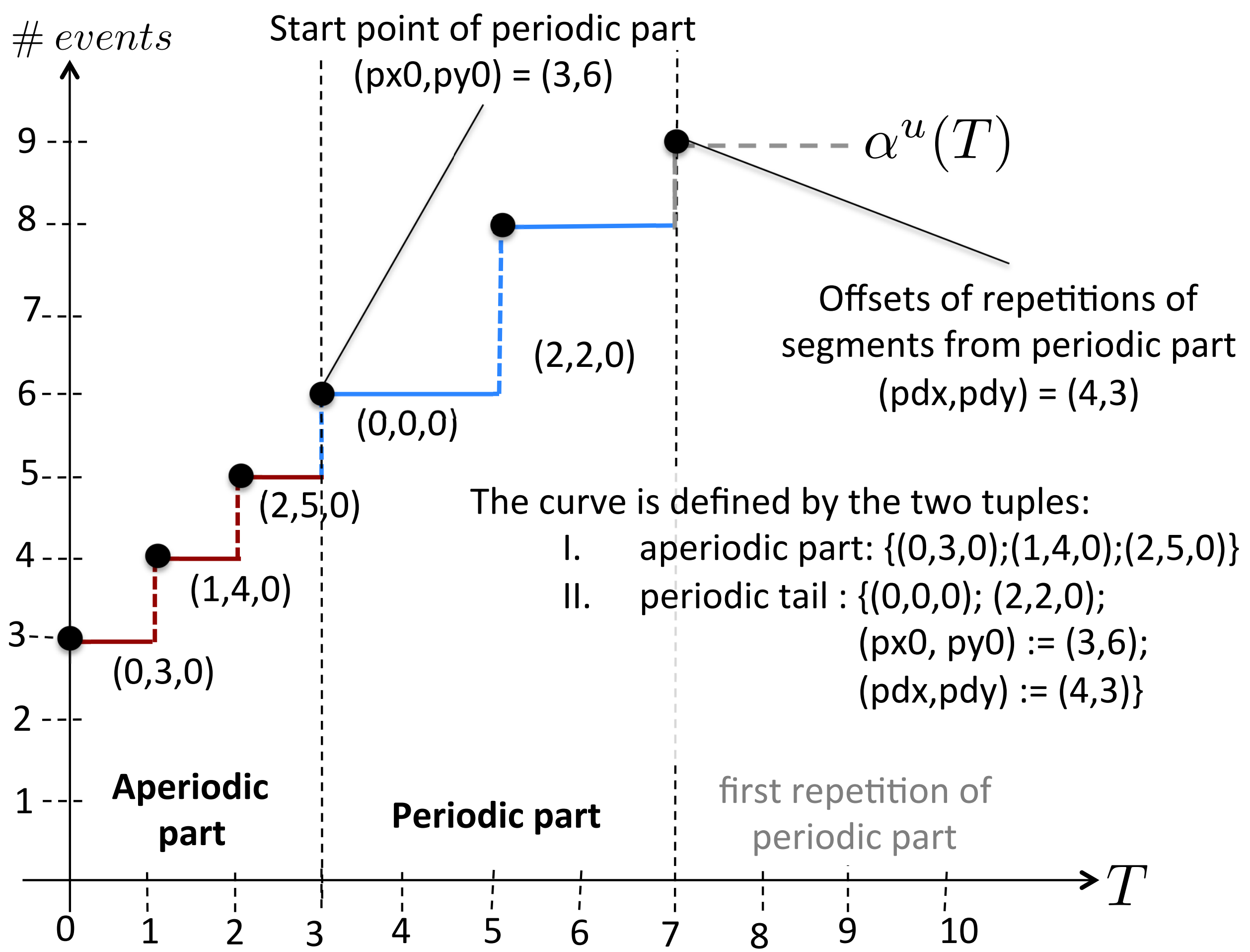
- Doing as less approximations as possible
  → acquiring the most accurate result

- Doing approximation automatically → providing a general framework instead of generating a use-case-specific solution

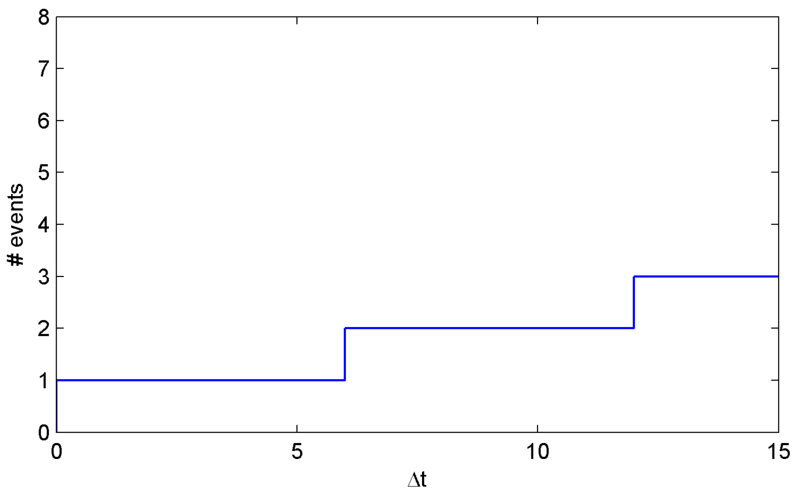
Goal: Analysis of the entire HCS reference topology

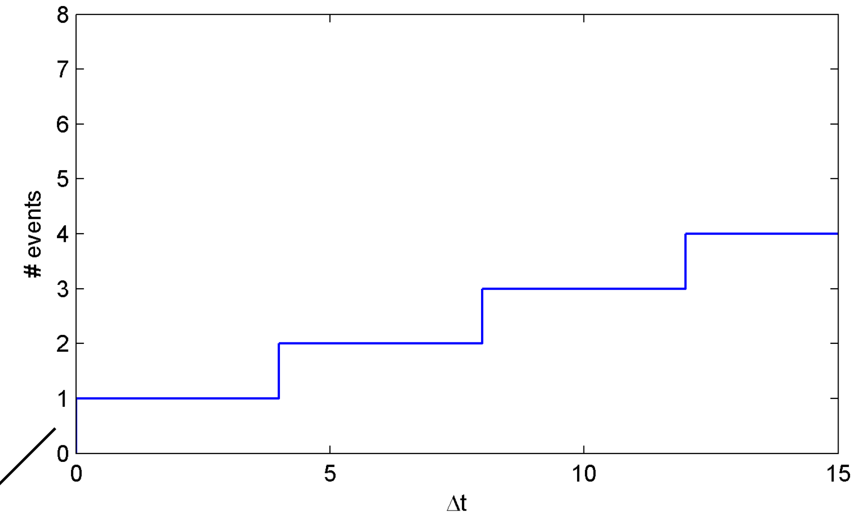
Source of the scaling problem

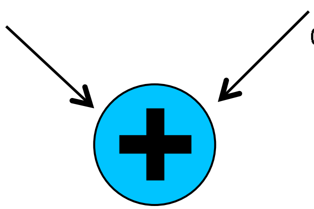
Many NC-based tools use pseudo-periodic curves, so does the MPA-toolbox



Source of the scaling problem

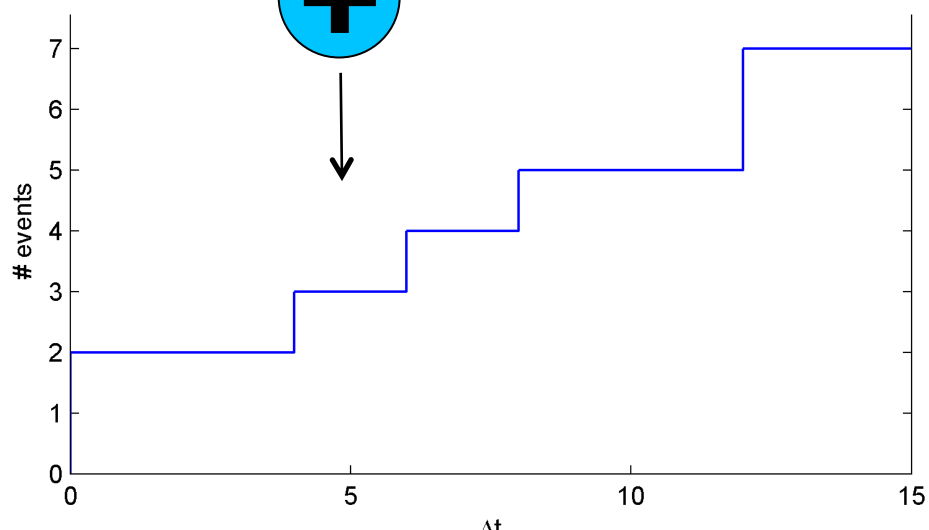






Stored segments of an output curve to a RTC-operation cover hyperperiod of input curves.

<u>Hyperperiod</u>:
least common multiple of periods of input curves.

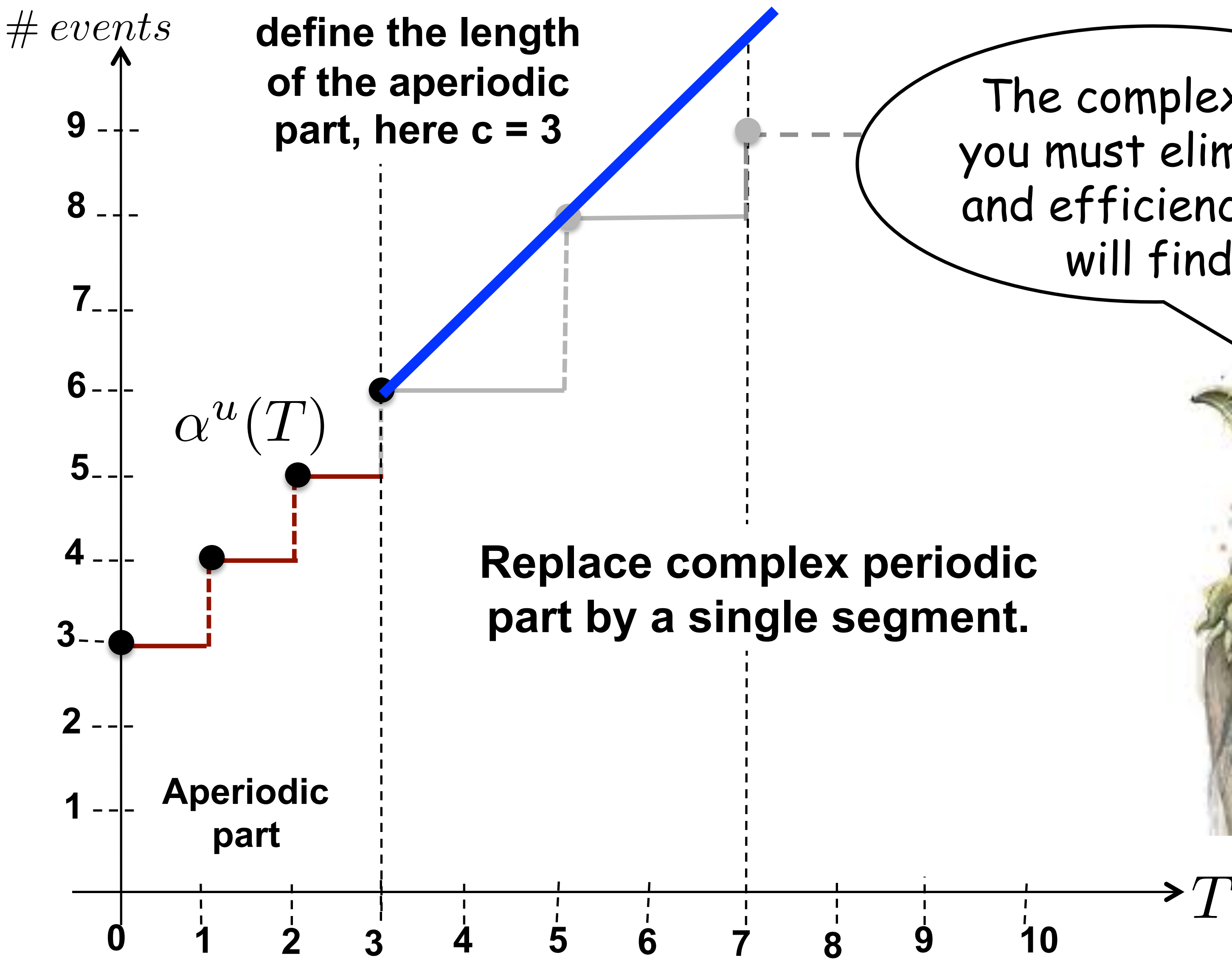


With co-prime periods of curves this yields an exponential blow up of curves

A first solution to the scaling problem

[Suppiger, Perathoner, Lampka, Thiele'10]

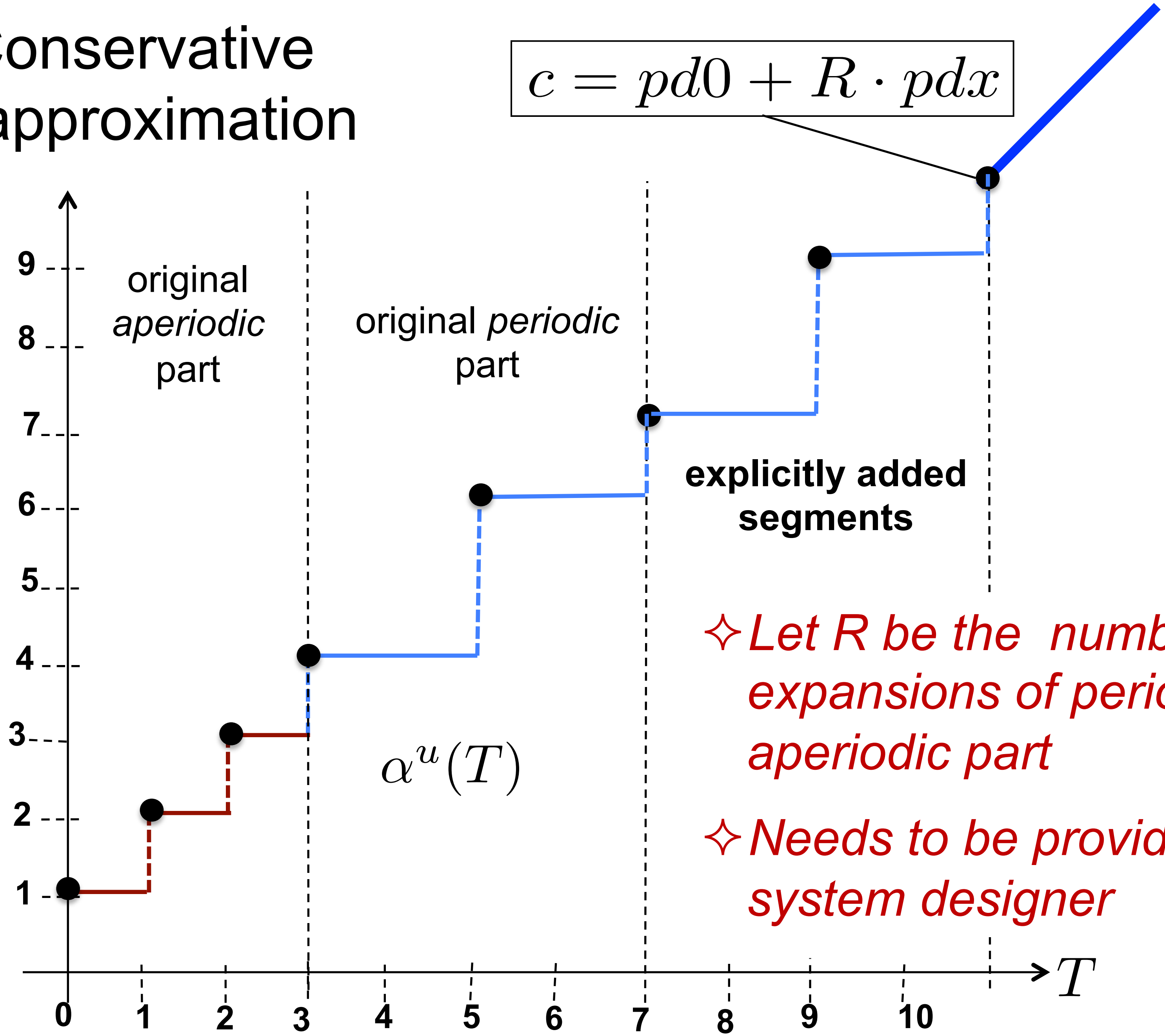
let constant c define the length of the aperiodic part, here c = 3



Replace complex periodic part by a single segment.

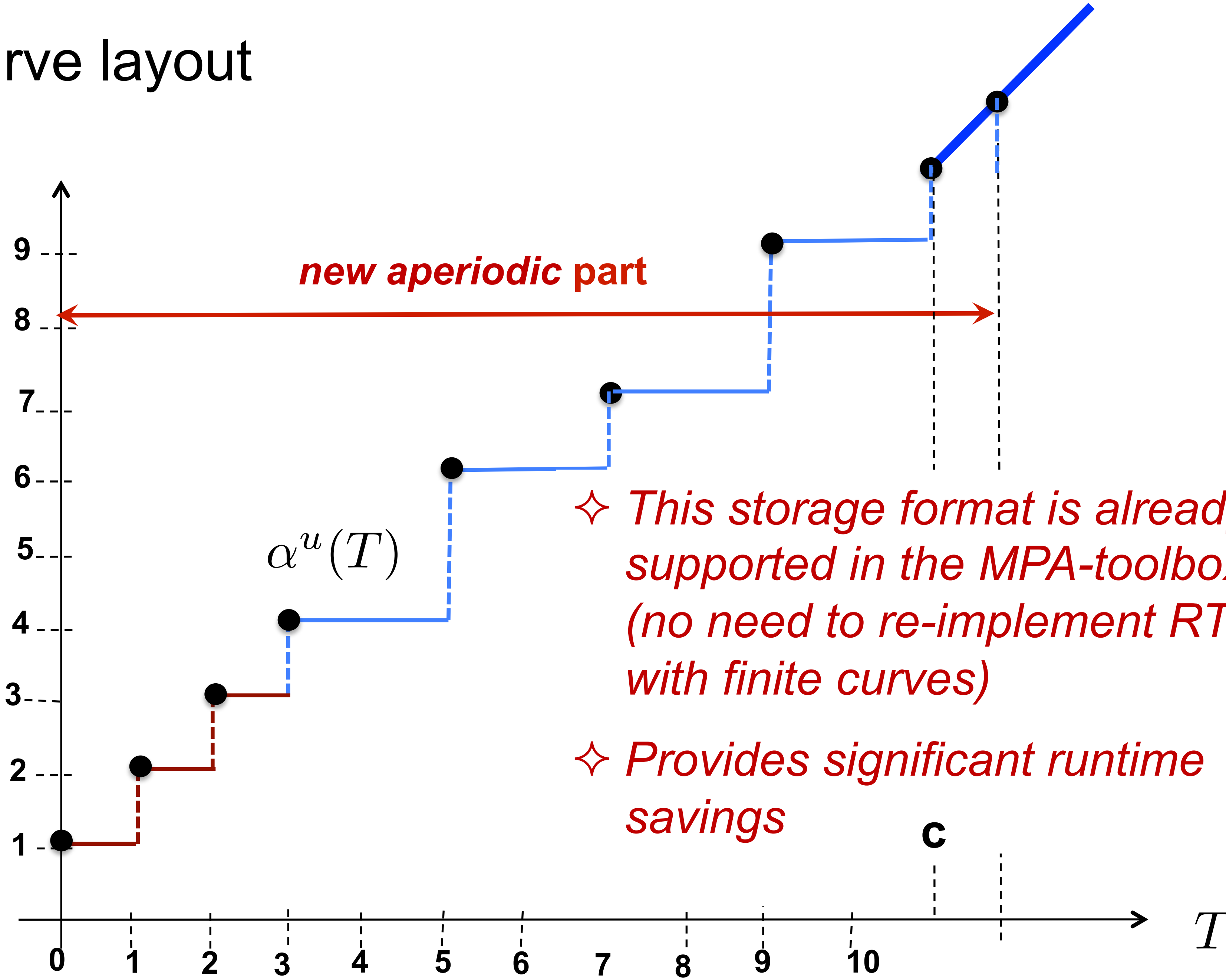
The complex tail you must eliminate and efficiency you will find.

Conservative approximation



- ✧ *Let R be the number of expansions of period into aperiodic part*
- ✧ *Needs to be provided by system designer*

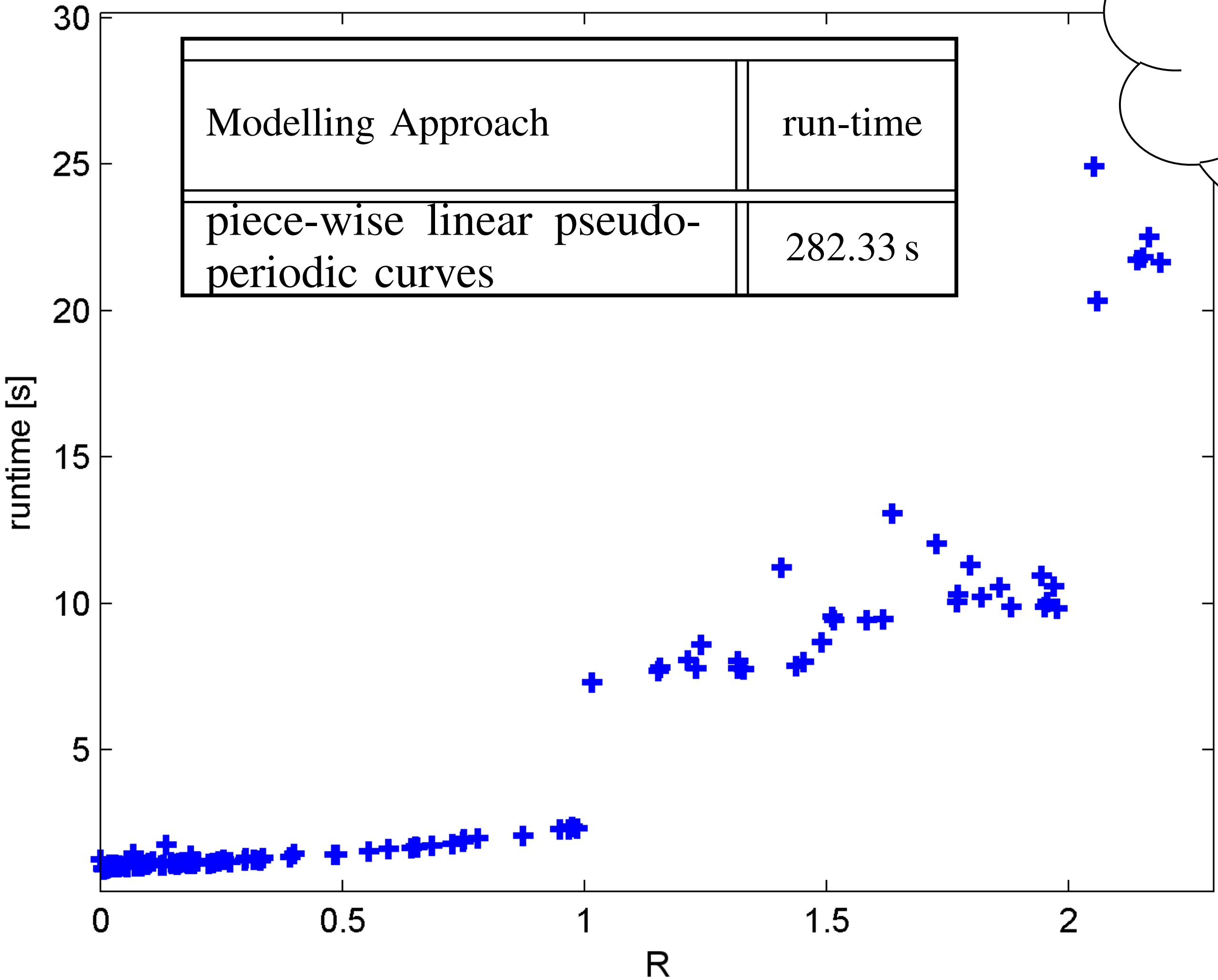
Curve layout



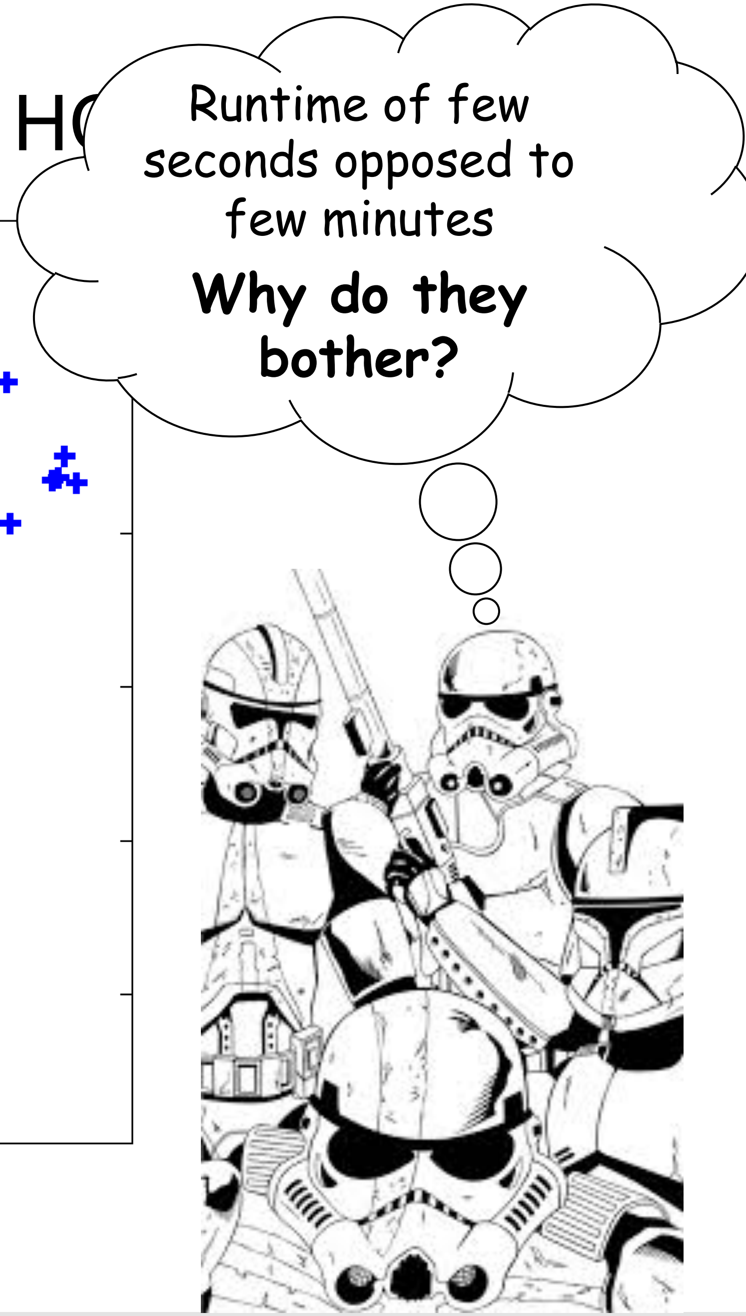
- ✧ *This storage format is already supported in the MPA-toolbox, (no need to re-implement RTC with finite curves)*
- ✧ *Provides significant runtime savings*

Use of conservative “conversion rule” (overapproximation) on the HC

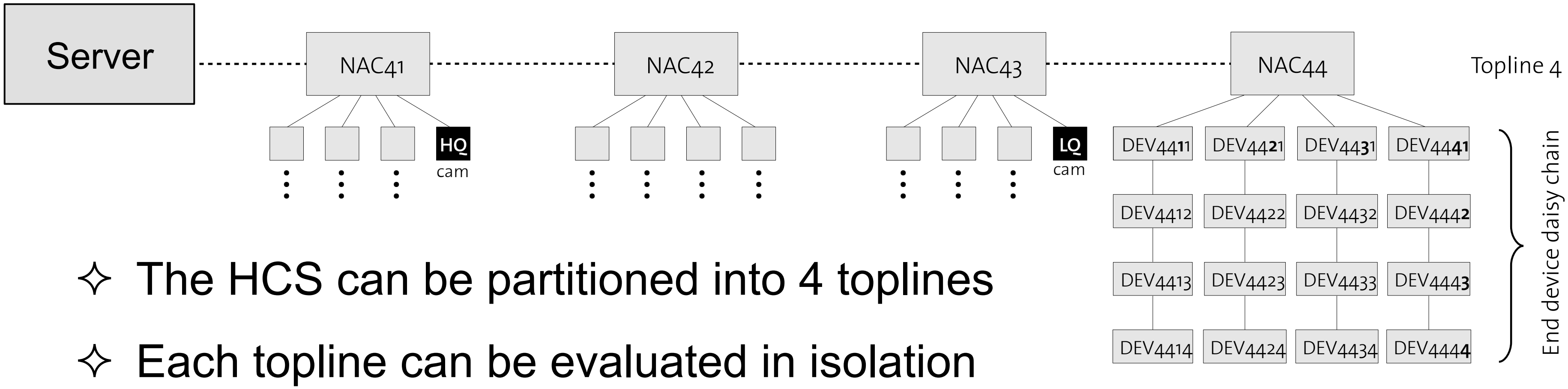
Modelling Approach	run-time
piece-wise linear pseudo-periodic curves	282.33 s



Effect on runtime as function of R



Moderate runtime of HCS is misleading:

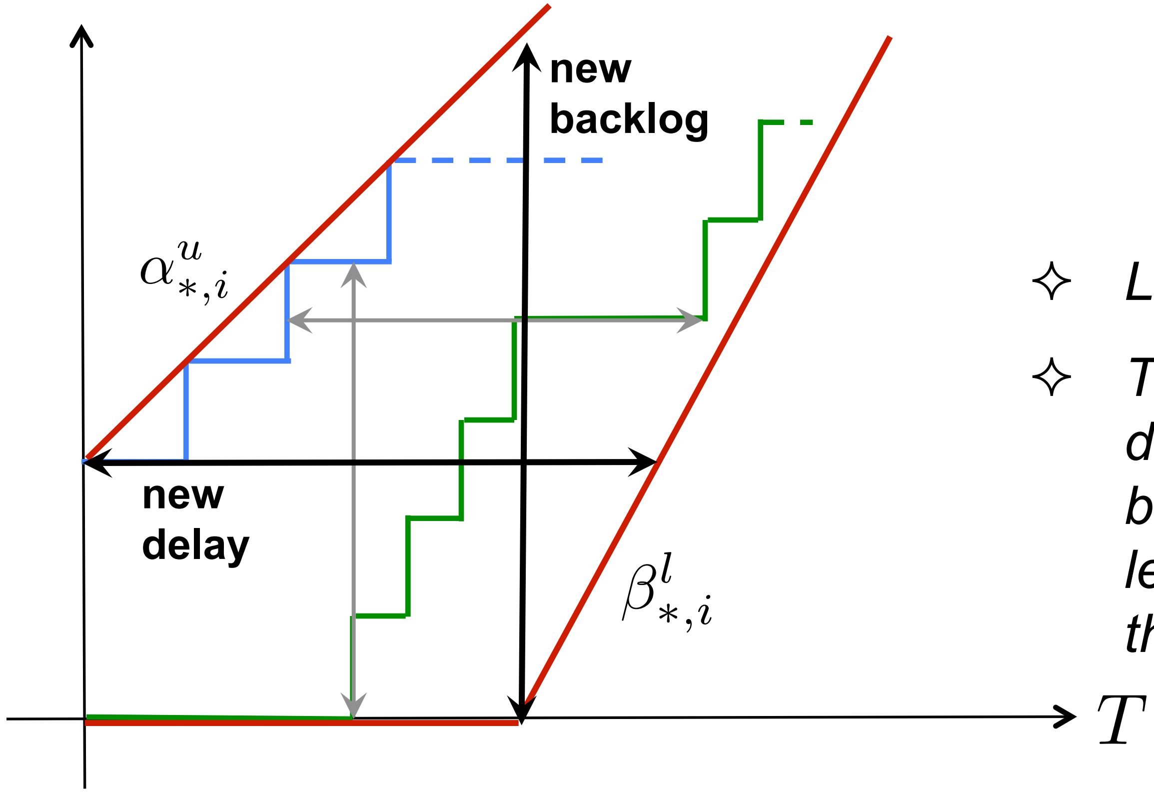


- ✧ The HCS can be partitioned into 4 toplines
- ✧ Each topline can be evaluated in isolation
- ✧ Upscaled resolution of curves (increases pessimism at the benefit of few number of segments)

Note: Quantitative evaluation methods might be part of design space exploration techniques

=> we need to be fast as possible

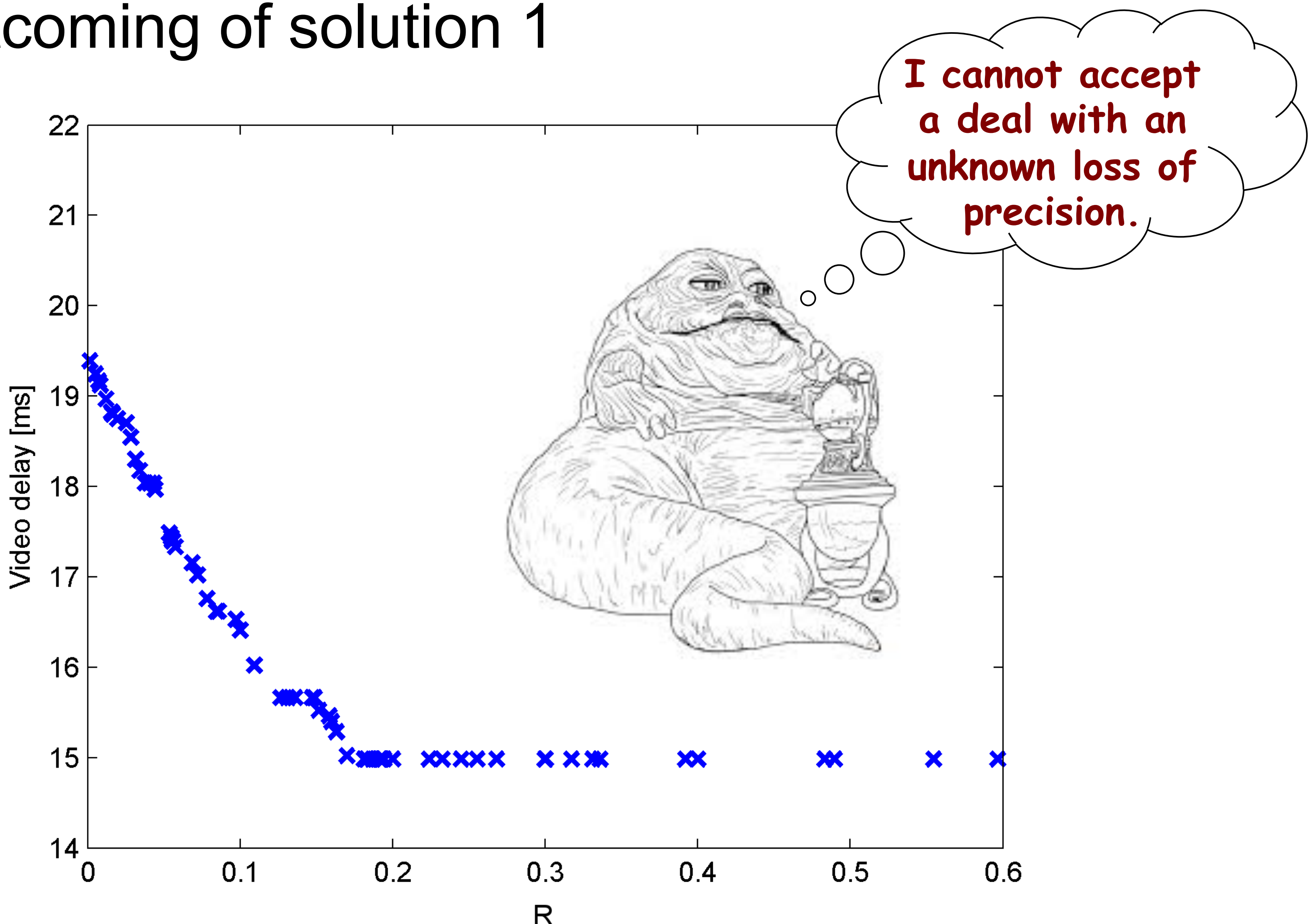
Shortcoming of solution 1



- *Losing accuracy.*
- *Tightness of the obtained delay/backlog bound becomes a function of the length of the prefix and thereby a function of R*

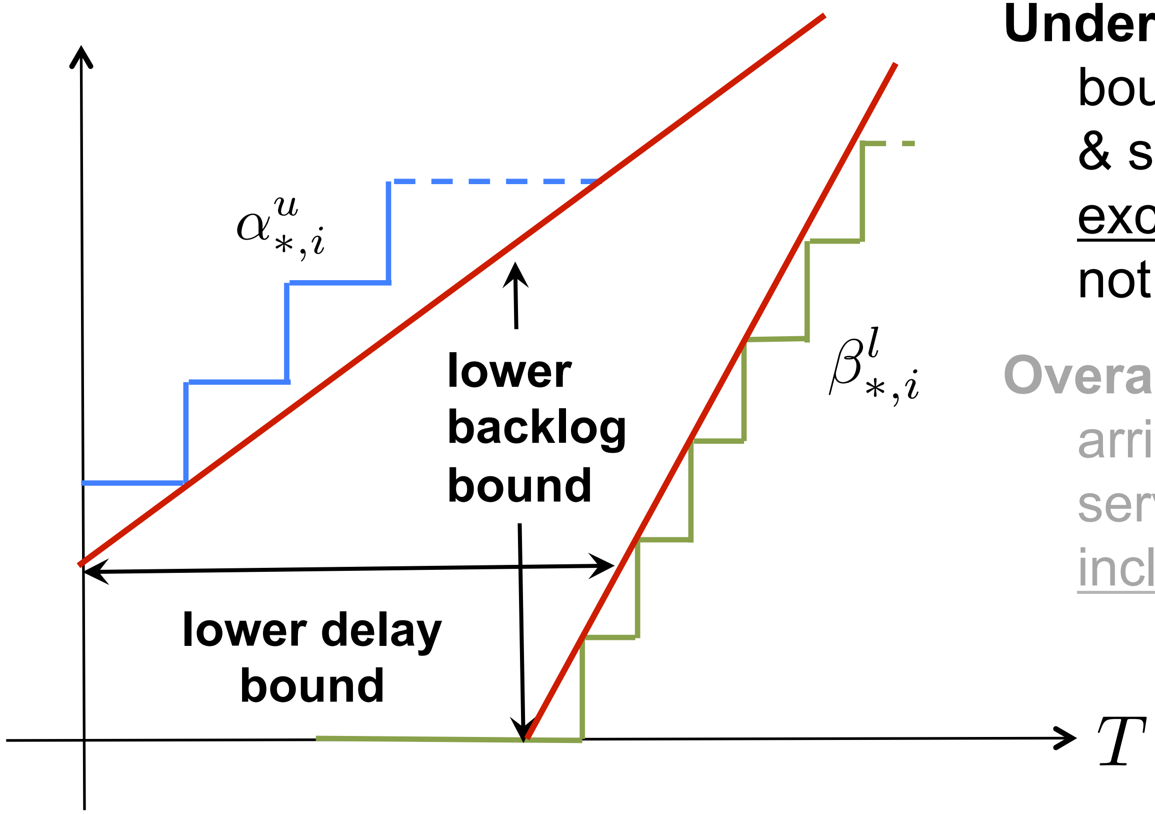
When done wrong (or arbitrary), unknown loss of tightness has to be expected.

Shortcoming of solution 1



End-to-end delay of the video stream in the HCS plotted for different values of R

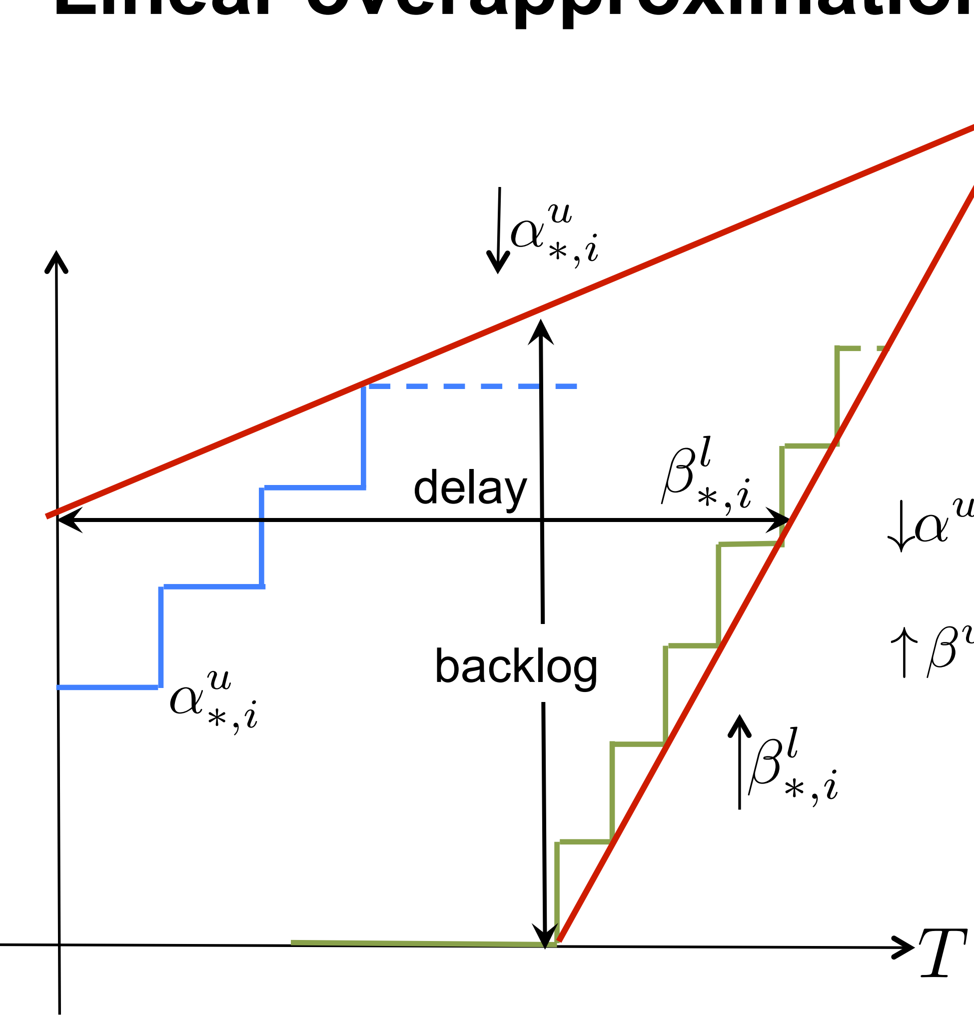
but......... at least we can bound the error by using linear “under-approximation”



Under-approximation:
bound arrival from below & service from above; excludes behaviour (is not safe)

Overapproximation: bound arrival from above & service from below; includes more behaviour

Linear overapproximation of arrival and service



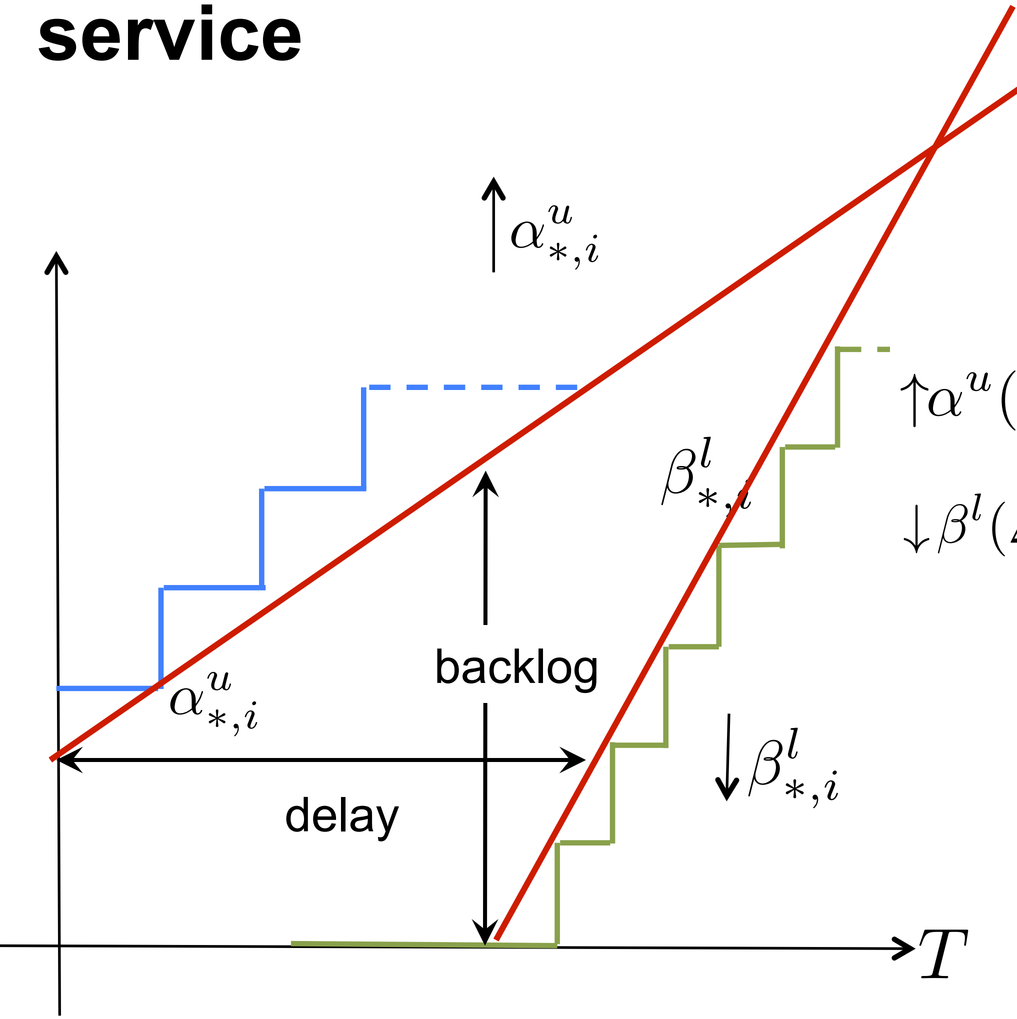
Overapproximation, linear bound from above and below:

$$\downarrow\alpha^u(\Delta) = \max(0, N_u + \rho \cdot \Delta) \geq \alpha^u(\Delta)$$

$$\uparrow\beta^u(\Delta) = \max(0, N_l + \gamma \cdot \Delta) \leq \beta^l(\Delta)$$

Backlog and delay bound derived from overapproximations are an upper bound on the actual values!

Linear under-approximations of arrival and service



Under-approximation, linear bound from below and above:

$$\uparrow\alpha^u(\Delta) = \max(0, N_u + \rho\cdot\Delta) \leq \alpha^u(\Delta)$$

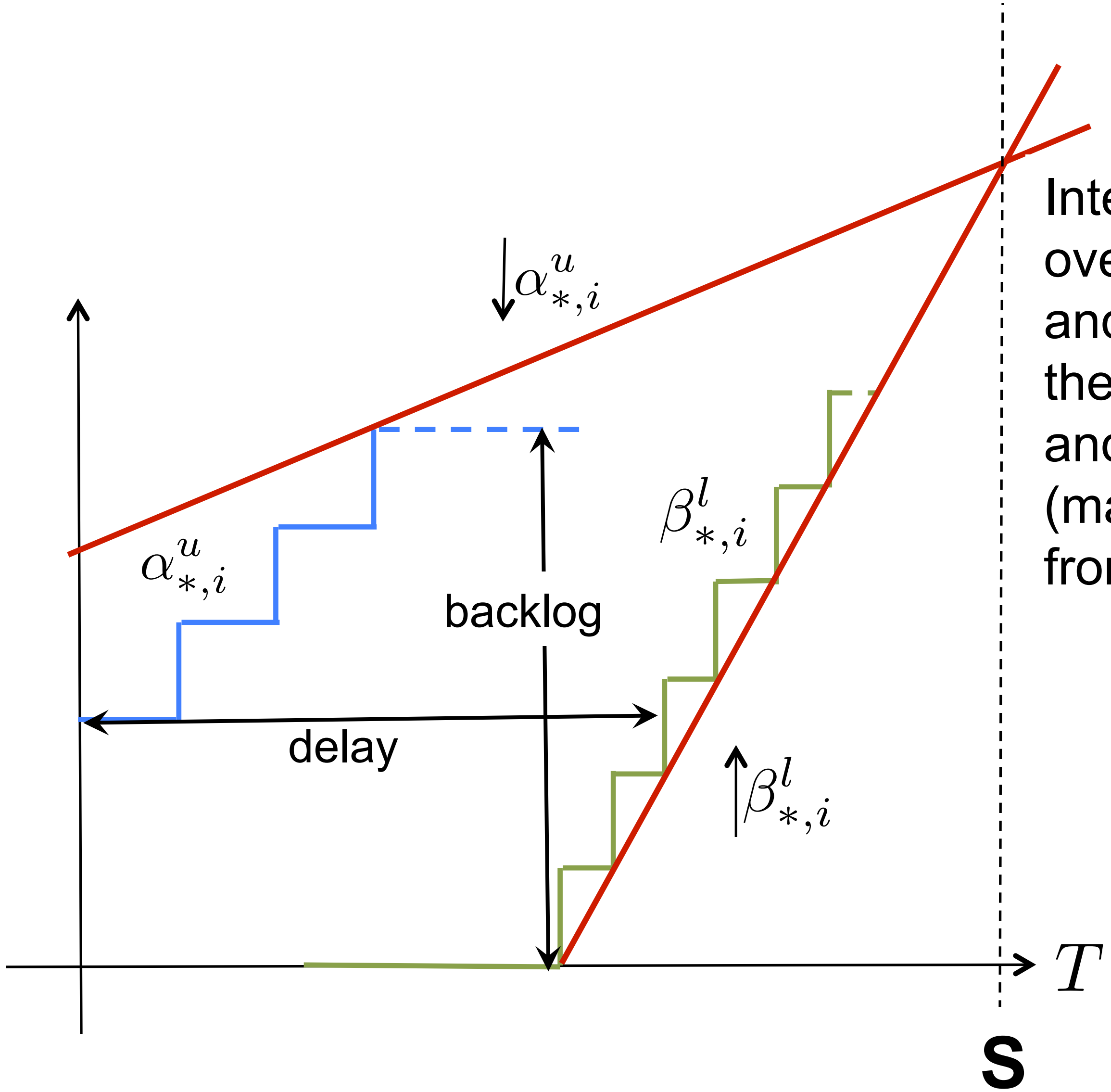
$$\downarrow\beta^l(\Delta) = \max(0, N_l + \gamma\cdot\Delta) \geq \beta^l(\Delta)$$

Backlog and delay bound derived from underapproximations are a lower bound on the actual values!

Take-away message for solution 1

- Linear overapproximation of periodic part (tail) speeds up computation significantly
- When done correctly, i..e, with an aperiodic part (prefix) of sufficient length no loss in precision has to be expected

Intuition for a rule to compute c for a component



Intersection of overapproximated arrival and service curves bounds the stretch where backlog and delay bound reside (max. busy window, known from scheduling theory)

Can this be a bound for c?

yes, but ....

$$(a \otimes b)(\Delta) = \inf_{0 \leq \lambda \leq \Delta}\{a(\Delta - \lambda) + b(\lambda)\}$$

$$\boxed{(a \oslash b)(\Delta) = \sup_{\lambda \geq 0}\{a(\Delta + \lambda) - b(\lambda)\}}$$

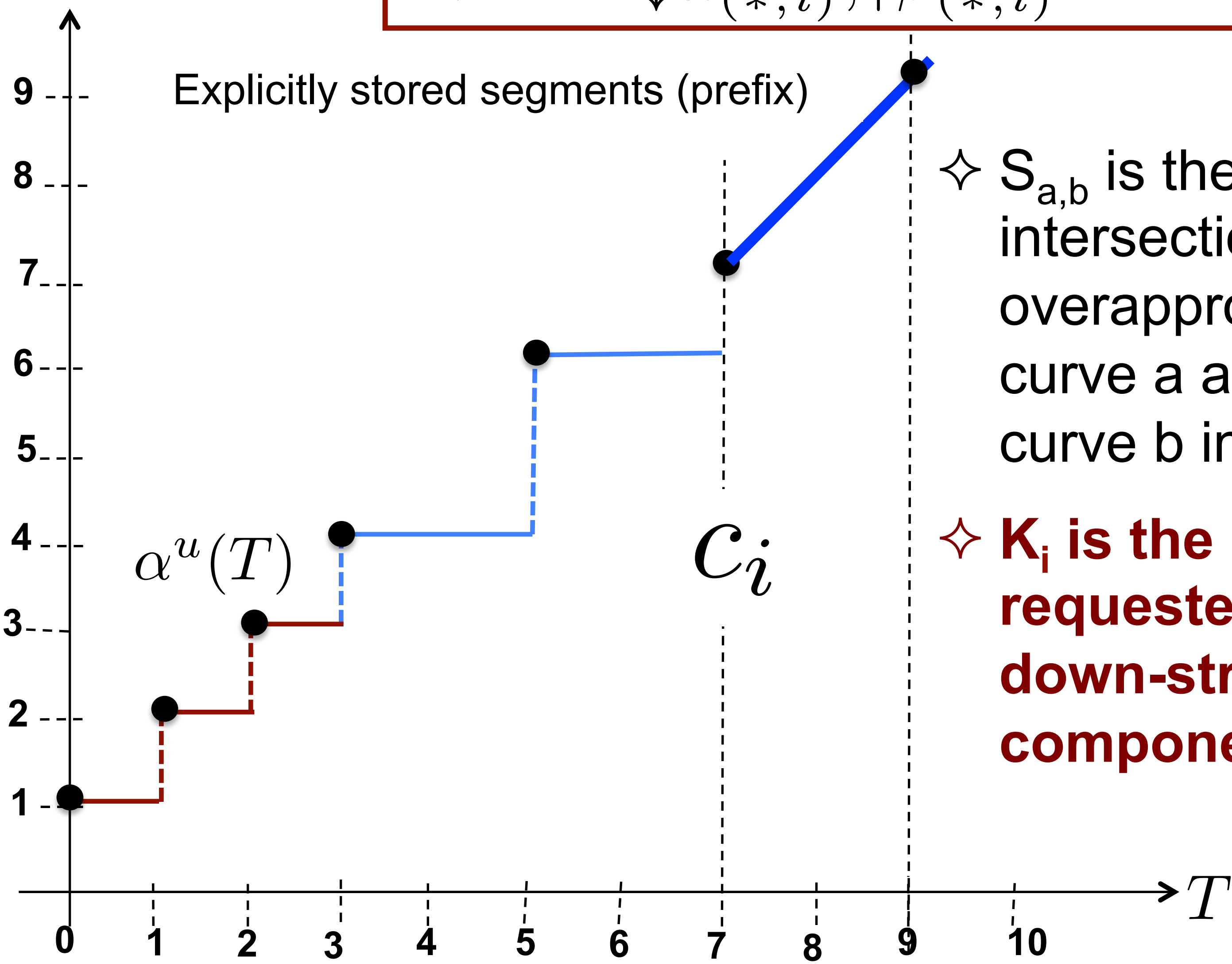
$$(a \,\overline{\otimes}\, b)(\Delta) = \sup_{0 \leq \lambda \leq \Delta}\{a(\Delta - \lambda) + b(\lambda)\}$$

$$\boxed{(a \,\overline{\oslash}\, b)(\Delta) = \inf_{\lambda \geq 0}\{a(\Delta + \lambda) - b(\lambda)\}}$$

- Convolution operations (Δ ≤ S implies 0 ≤ λ ≤ S)
  If the output curve needs to be defined up to S, input curve a and b have to be defined only up to S too (problem solved).
- Deconvolution operations (Δ ≤ S implies ???)
  No trivial solution as no bound for λ can be derived from the definitions directly.

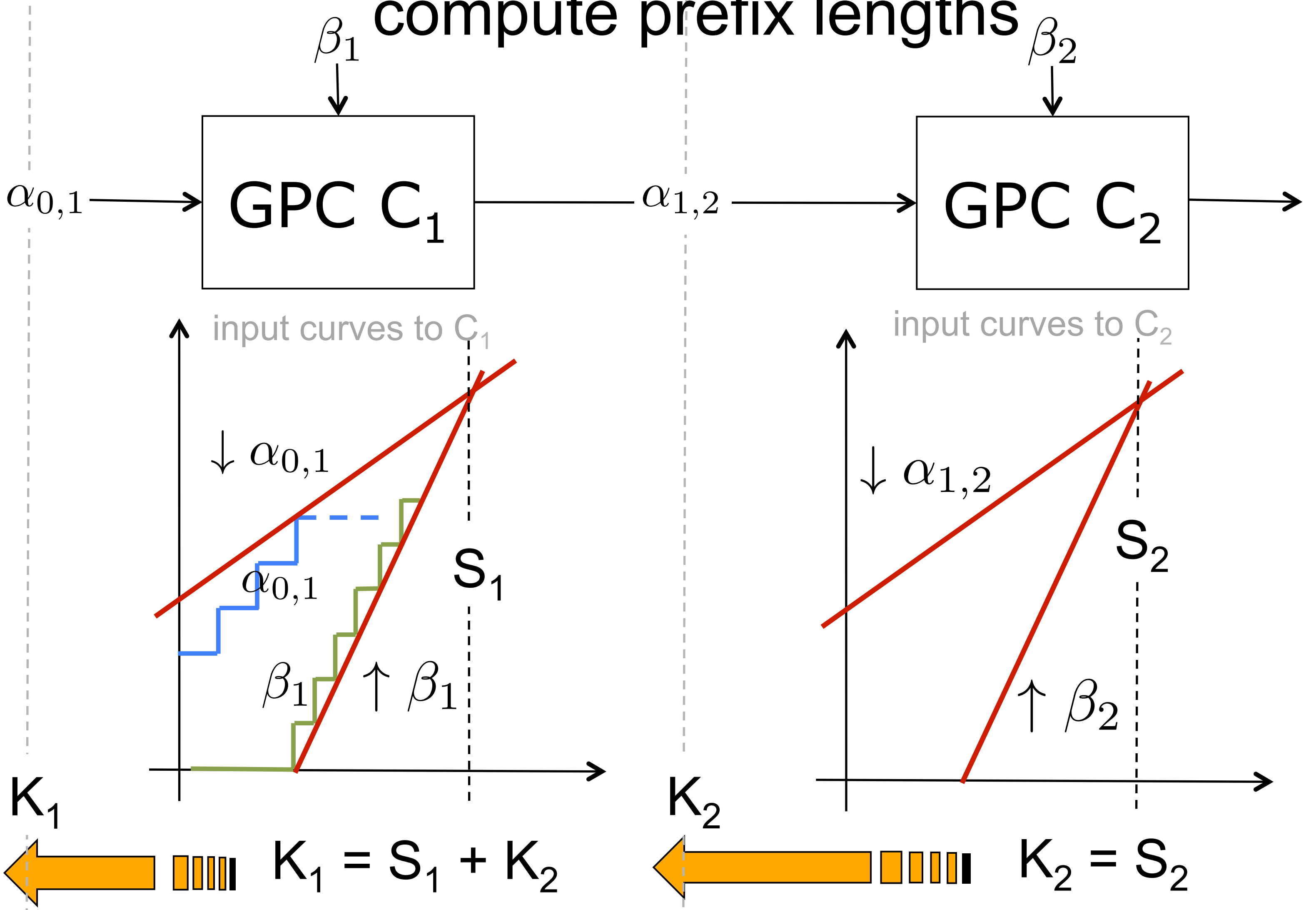
A second (GPC-based) solution [Guan,Wang'13]

$$\boxed{c_i = S_{\downarrow\alpha_{(*,i)},\uparrow\beta_{(*,i)}} + K_i}$$



- $S_{a,b}$ is the latest intersection value of overapproximated arrival curve a and service curve b input to GPC C_i
- **K_i is the prefix size requested by the down-streamed component of C_i**

Execute analysis with overapproximations and compute prefix lengths



propagation of prefix size along the transitive closure of the input relation

Take away message for 2nd solution

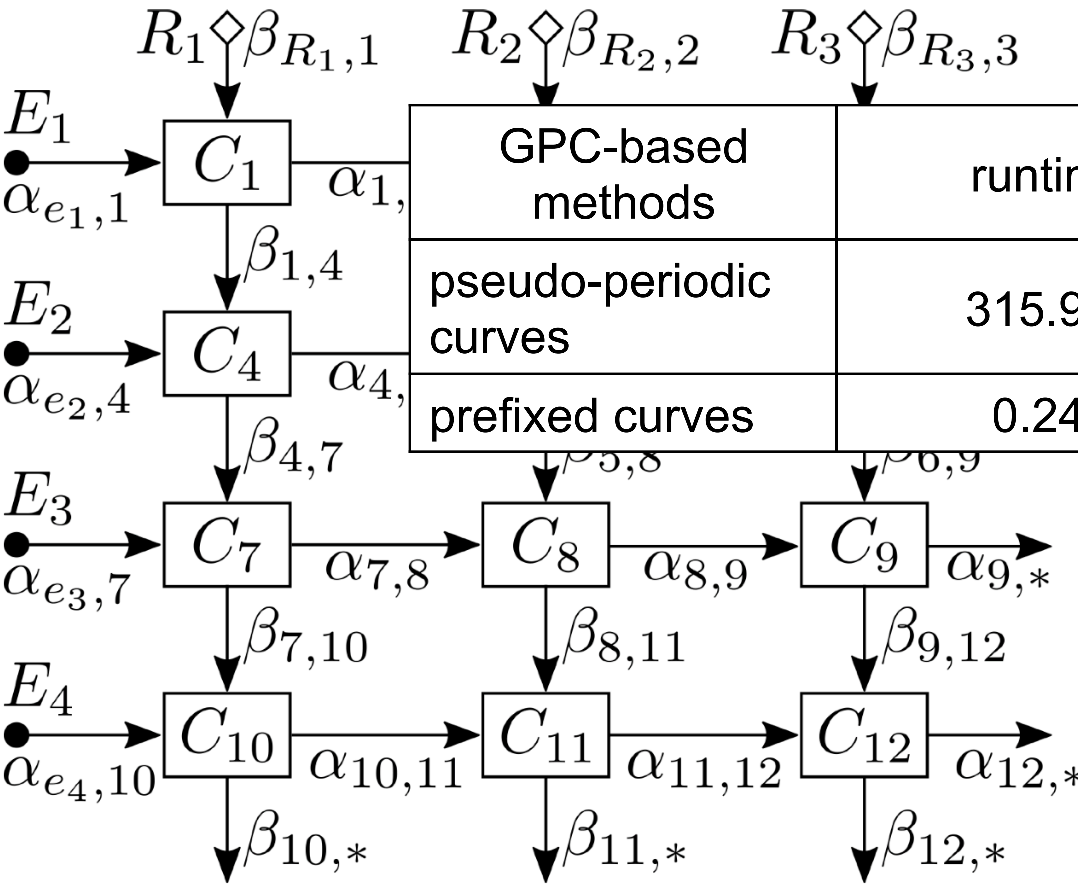
① Computation of prefix sizes can be done efficiently

- ✧ System analysis based on linear overapproximations
- ✧ back-propagation of prefix sizes along the input paths

② Size of prefixes resembles “busy window” approach known from scheduling theory **and proofs related to the GPC** and its input curves. **How to be used with other “component models” or NC-theorem like PBOO, PMOO was unknown.**

③ GPC-based system analysis does not reflect the state-of-the-art (to slow & not tight)

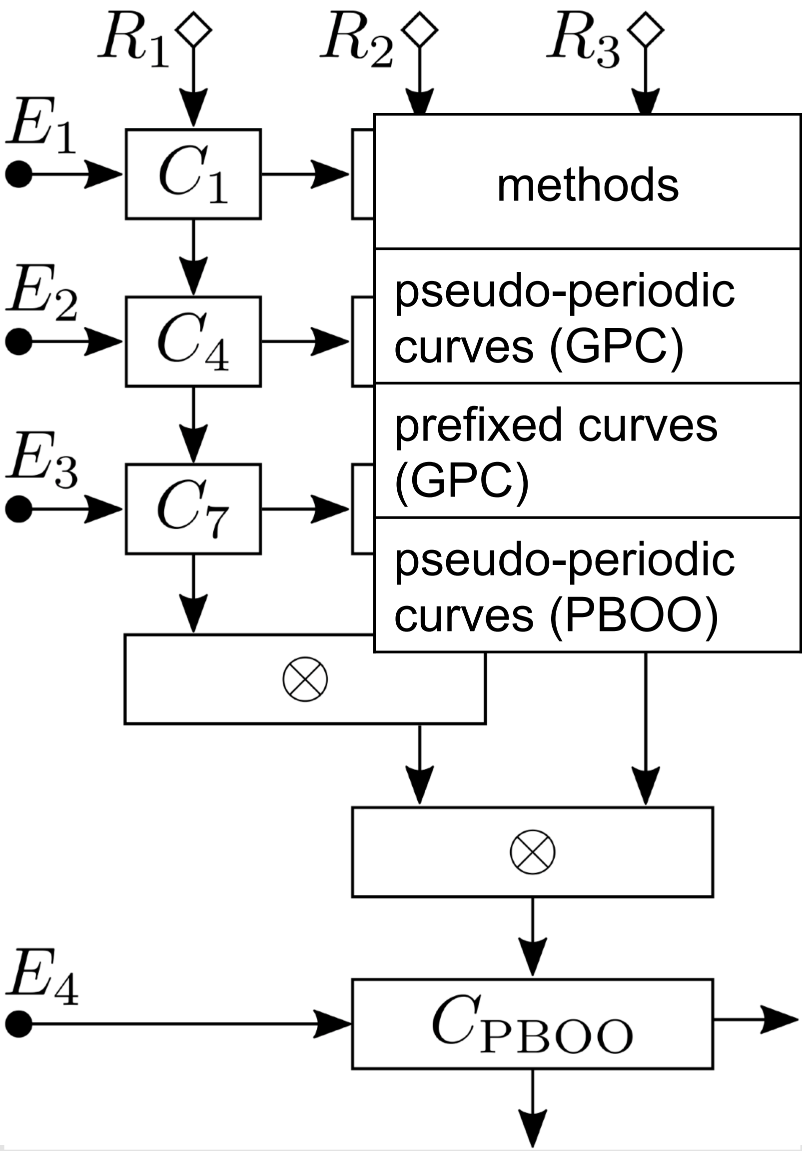
Running example from [Guan,Wang'13]



GPC-based methods	runtime	end-to-end delay E_4
pseudo-periodic curves	315.97s	21
prefixed curves	0.24s	21

Tightness of results

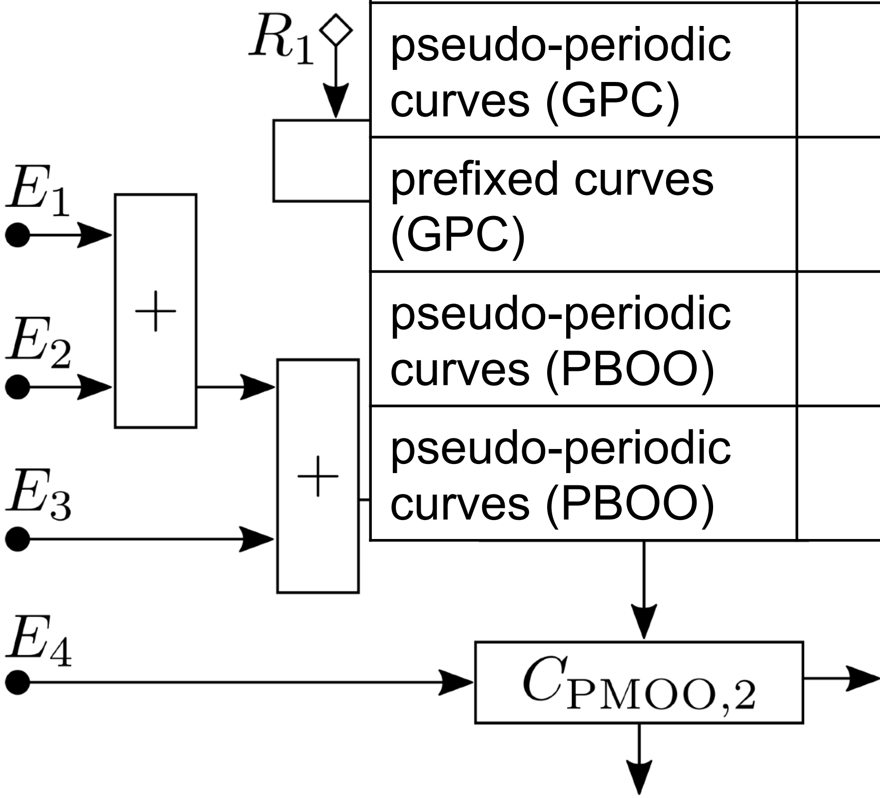
Exploiting PBOO with running example



methods	runtime	end-to-end delay E_4
pseudo-periodic curves (GPC)	315.97s	21
prefixed curves (GPC)	0.24s	21
pseudo-periodic curves (PBOO)	35.67s	19

Can we do still better

Exploiting PMOO with running example

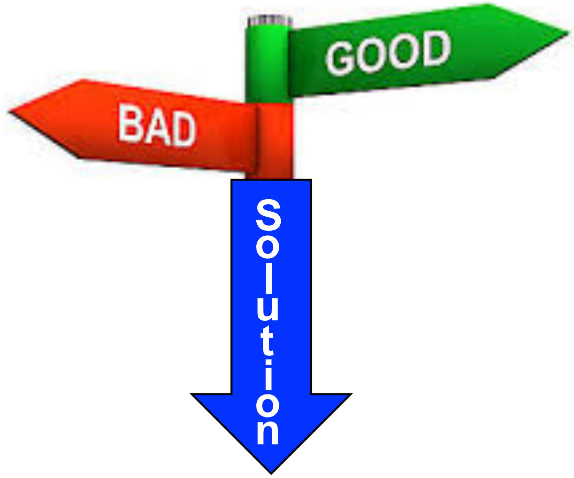


methods	runtime	end-to-end delay E_4
pseudo-periodic curves (GPC)	315.97s	21
prefixed curves (GPC)	0.24s	21
pseudo-periodic curves (PBOO)	35.67s	19
pseudo-periodic curves (PBOO)	17.67s	15

Towards a new solution

(GPC-based)
Finitary-RTC

with GPC-style of analysis we significantly lose precision



with prefixed curves we gain significantly speed in the analysis

Finitary-RTC but with prefix sizes derived for RTC-operators (and not GPCs)

.......also simplifies proofs

Problem definition:

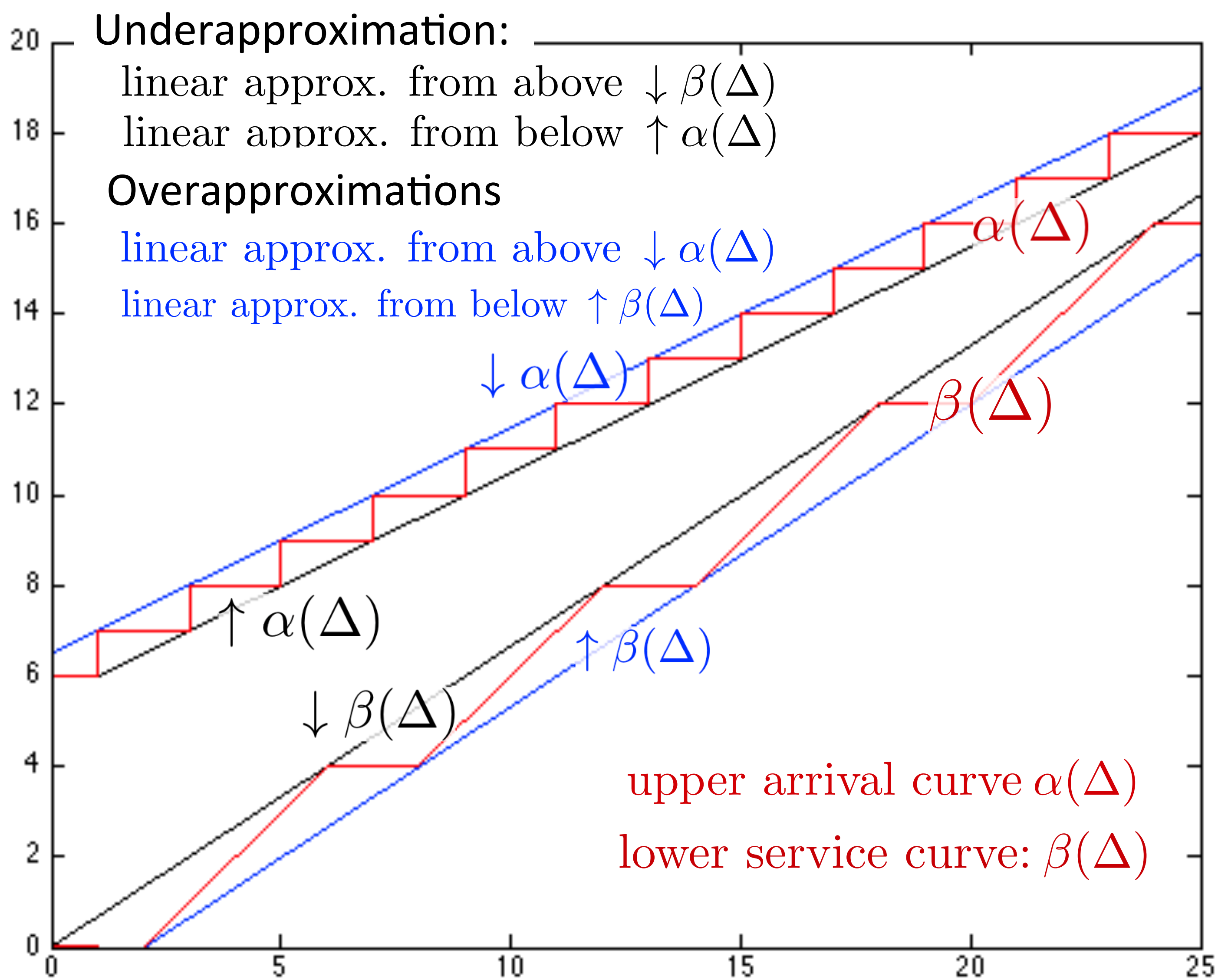
- Let curves a and b be standard RTC curves defined on the interval $[0, +\infty)$.
- Let curves a' and b' be their prefixed counterparts, i.e., they are defined on the finite interval $[0, k_a]$ and $[0, k_b]$, where for $\Delta \in [0, k_a] : a(\Delta) = a'(\Delta)$ and for $\Delta \in [0, k_b] : b(\Delta) = b'(\Delta)$ holds.
- Let $\odot$ be an operator relevant for RTC, i.e.,

$$\odot \in \{\otimes, \overline{\otimes}, \min, +, -, \oslash, \overline{\oslash}\}$$

For a finite constant k we need to clarify the condition on the size of k_a and k_b w.r.t.k and operator $\odot$ such that

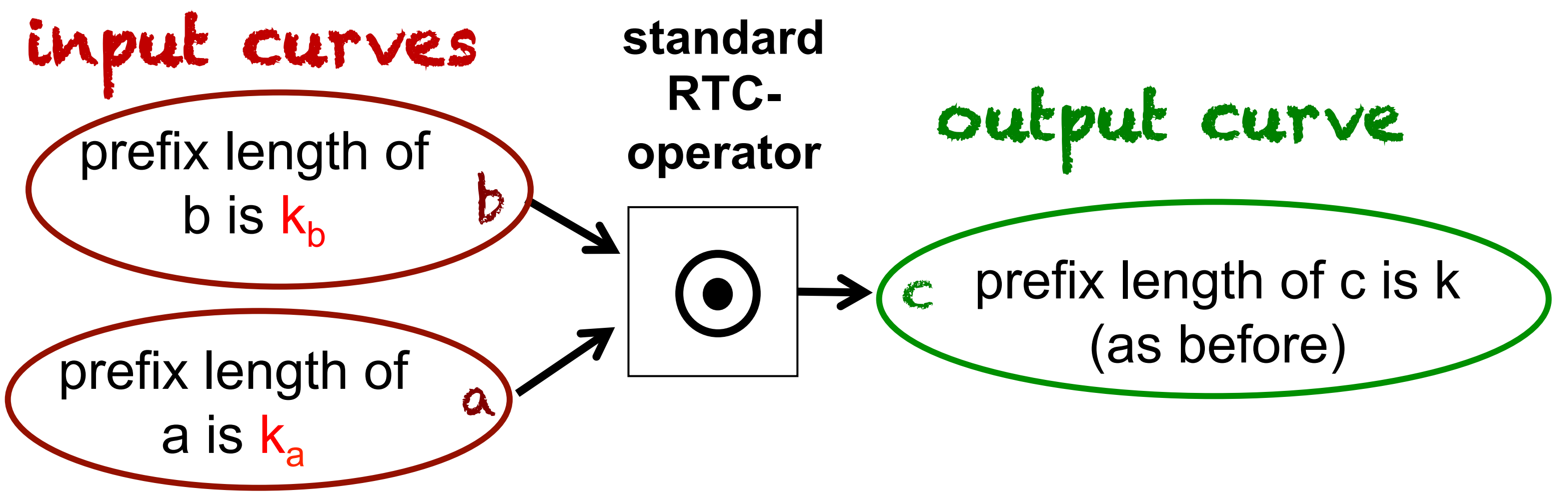
$$\boxed{\forall \Delta \in [0, k] : \quad (a \odot b)(\Delta) = (a' \odot b')(\Delta)}$$

Recall: linear over- and underapproximations as defined before:



Domain bounds with common RTC operators

Let the operators of group I be from the set $\{\otimes, \overline{\otimes}, \min, +, -\}$.



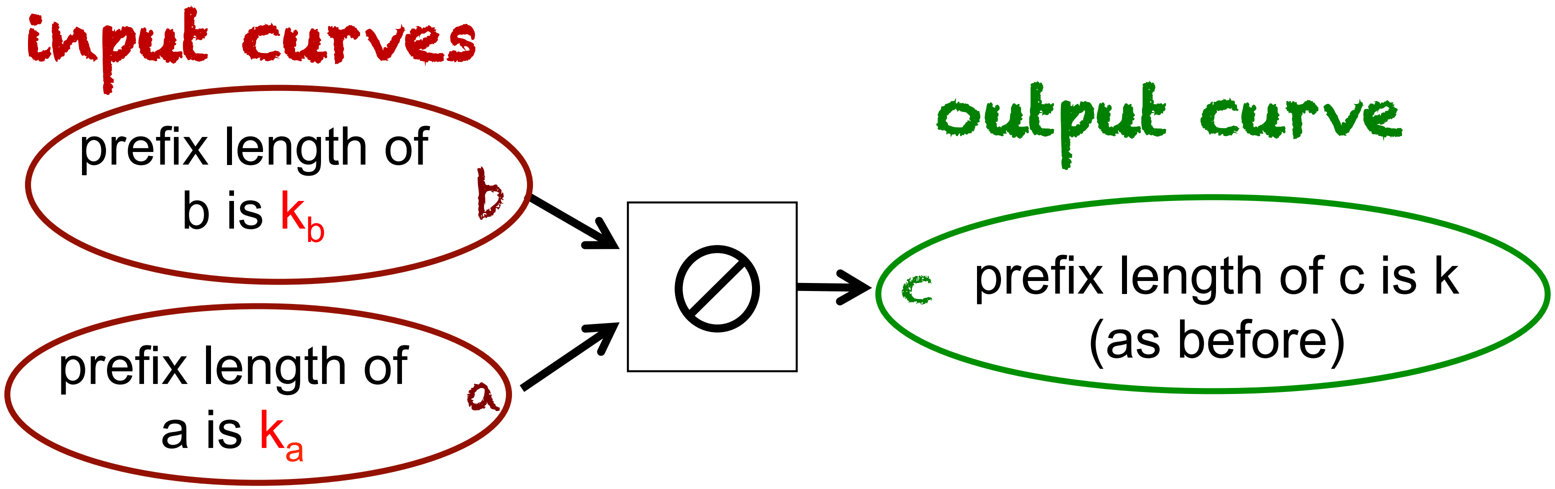
Satisfaction of the relation $k \leq \min(k_a, k_b)$ yields that for $\odot \in \{\otimes, \overline{\otimes}, \min, +, -\}$ and $\Delta \in [0, k]$:

$$(a \odot b)(\Delta) = (a' \odot b')(\Delta) \text{ holds} \qquad .$$

This directly arises from the definition of these RTC-operators, e.g.,

$$(a \otimes b)(\Delta) = \inf_{0 \leq \lambda \leq \Delta}\{a(\Delta - \lambda) + b(\lambda)\}$$

Domain bounds with min-plus deconvolution



There is no trivial solution, because:

$$(a \oslash b)(\Delta) = \sup_{\lambda > 0}\{a(\Delta + \lambda) - b(\lambda)\}$$

but ....

Recipe for deriving domain bound for min-plus deconv.

① Compute lower bound on maximum vertical distance of a and b (backlog bound with underapproximations)

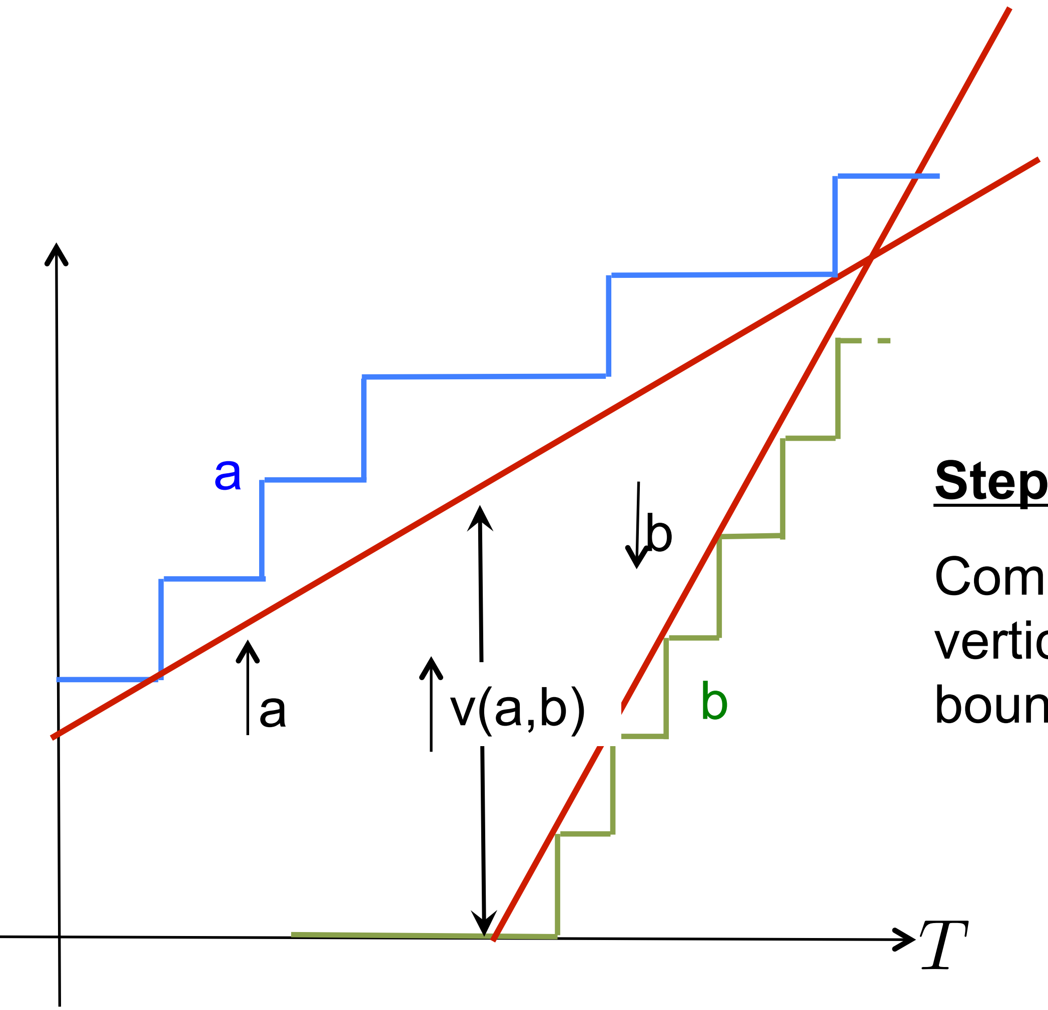
② Compute pseudo-inverse **vdp** of the lower vertical distance bound but with respect to **over-approximations** of a and b.

Assuming that **a is subadditive, b superadditive** and their **longterm rates are not equal, i.e.,** the backlog bound is finite

yields following property

$$\sup_{0 \leq \lambda \leq vdp} \{a(\Delta + \lambda) - b(\lambda)\} \geq \sup_{\lambda > vdp} \{a(\Delta + \lambda) - b(\lambda)\}$$

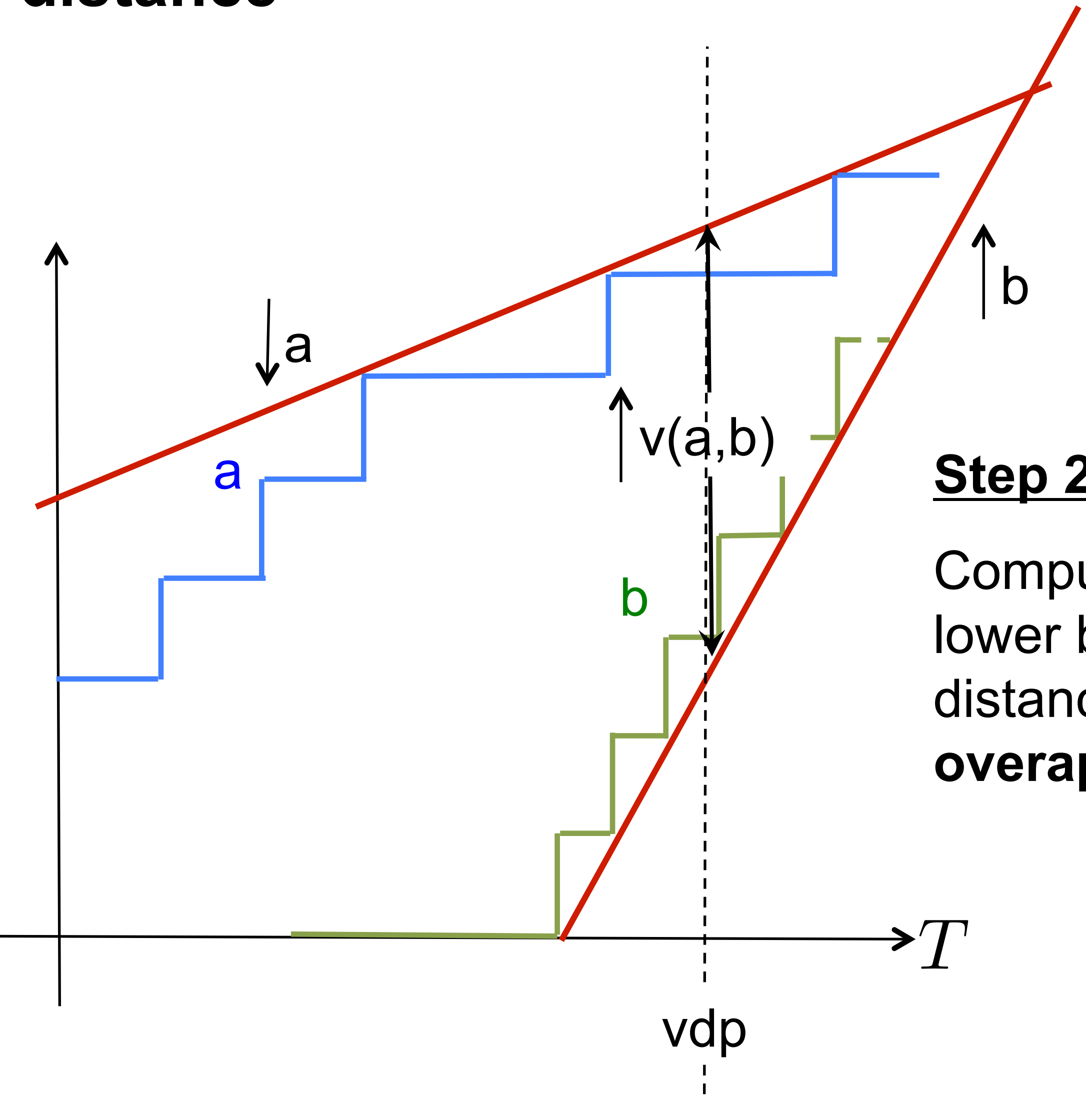
Lower bounding the max. vertical distance



Step 1

Compute lower bound on maximum vertical distance of a and b (backlog bound with **underapproximations**)

Pseudo-inverse of the lower bound on the max. vertical distance

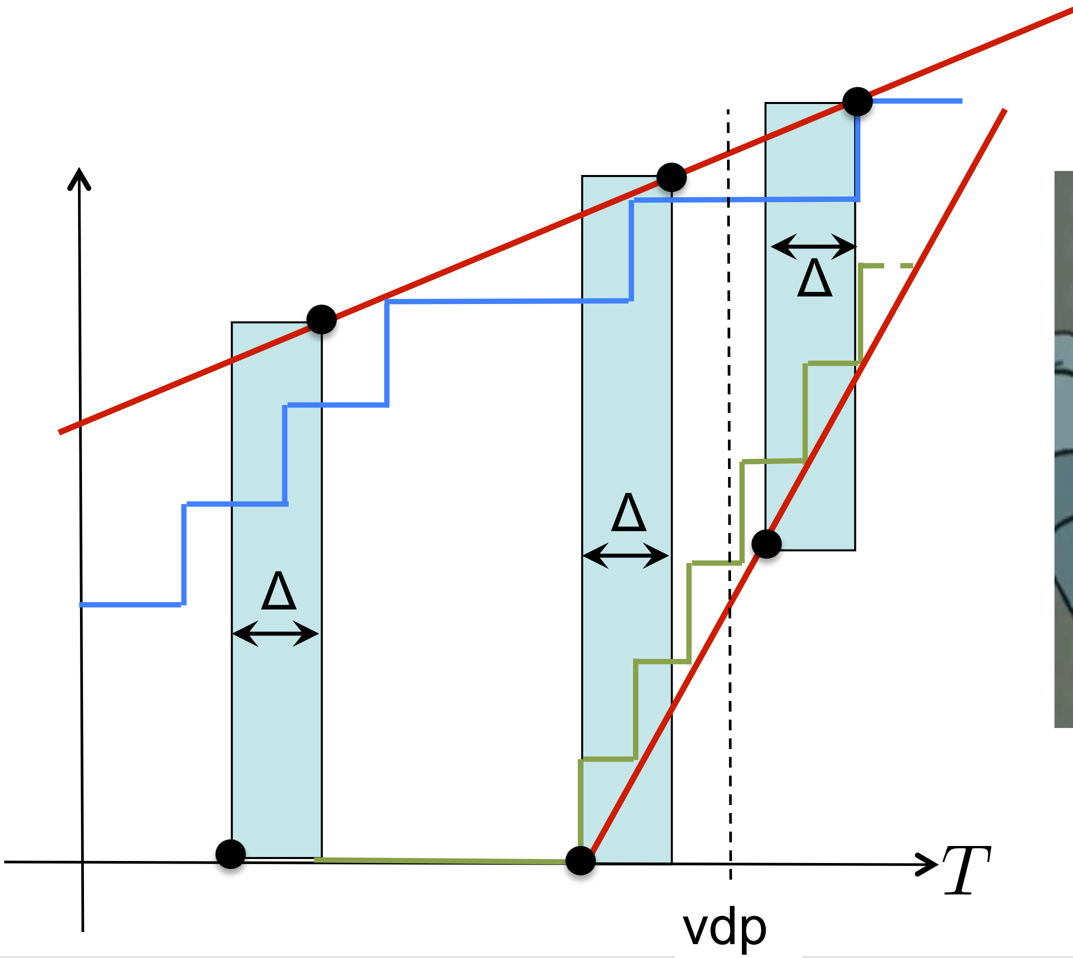


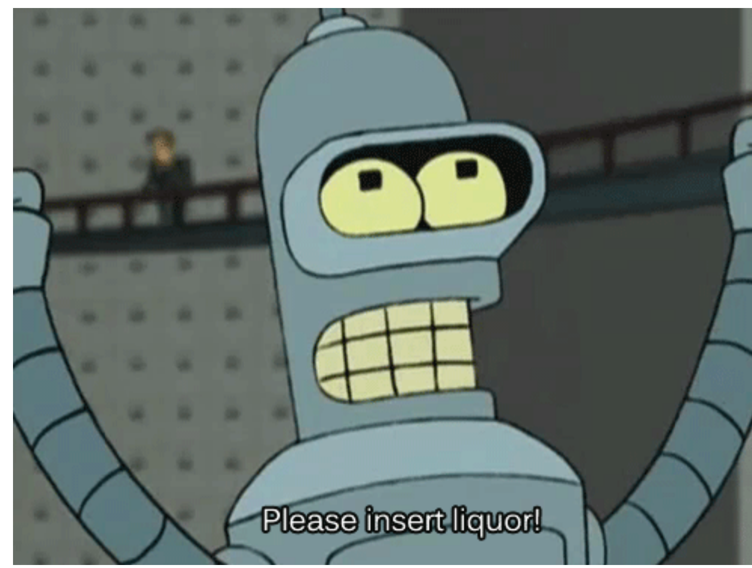
Step 2

Compute pseudo-inverse vdp of lower bound on maximum vertical distance **but w.r.t. overapproximations of a and b.**

When computing the min-plus deconvolution for a specific Δ

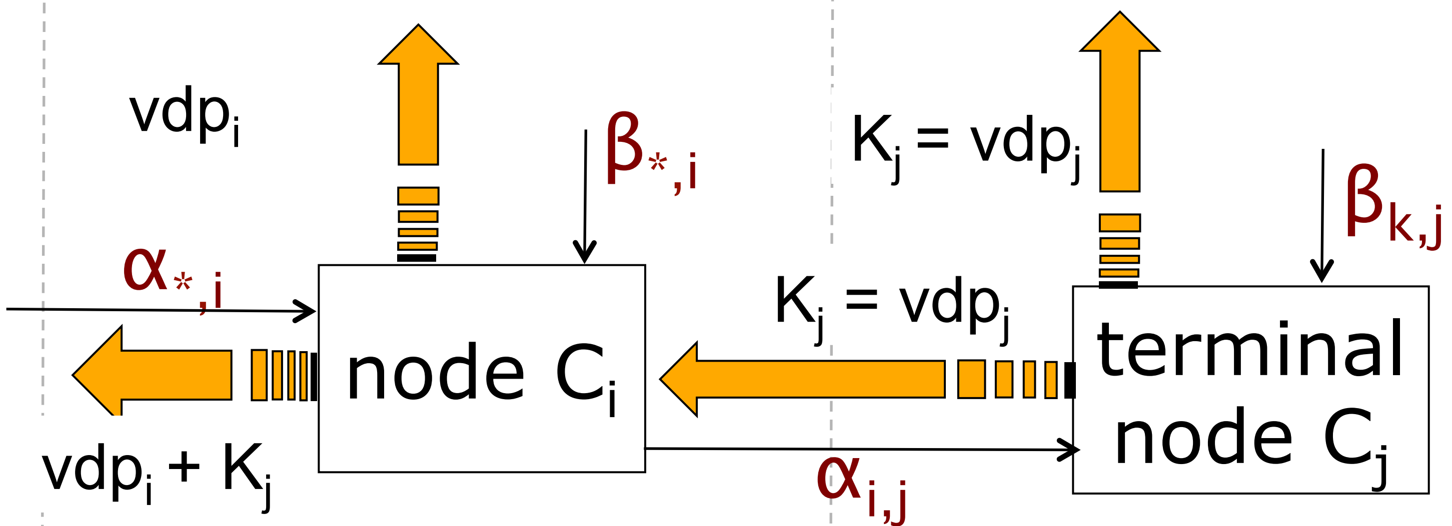
$$\sup_{0\leq\lambda\leq vdp}\{a(\Delta+\lambda)-b(\lambda)\} \geq \sup_{\lambda>vdp}\{a(\Delta+\lambda)-b(\lambda)\}$$





yeah we are done!
almost....

Execute analysis with over- and underapproximations to compute prefix lengths



- ✧ At an inner node C_i we (back)propagate (the max.) vdp_i in the direction of $\alpha_{*,i}$ and $vdp_i + K_j$ for $\beta_{*,i}$
- ✧ At the terminal node of a path, it is sufficient to back-propagate the max. vdp of that node.

For the running example from [Guan,Wang'13] Runtimes

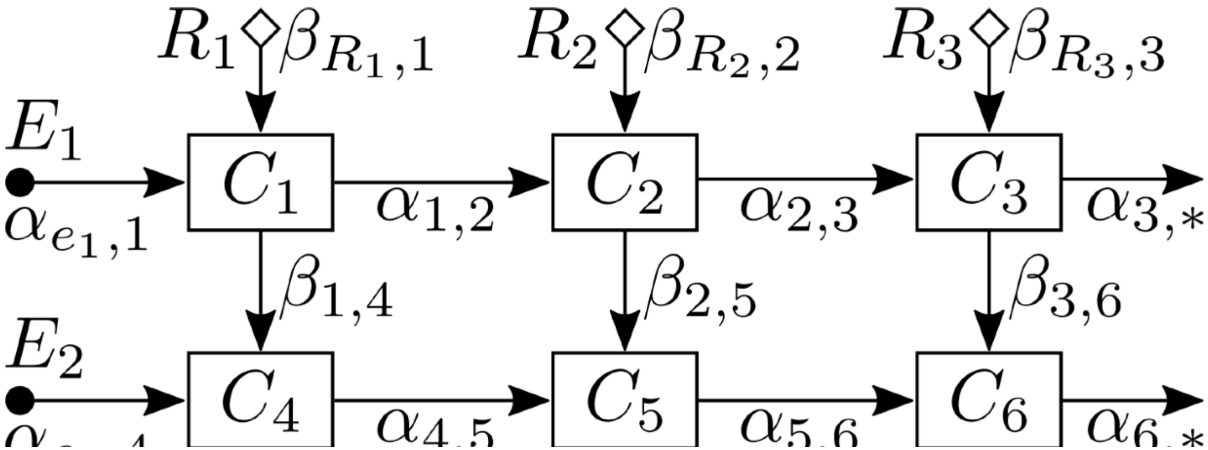
(a) Run times and delay bounds

	Linear	RTC	C_i-finit	$\odot$-finit
GPC	0.137s	315.97s	0.240s	0.228s
PBOO	0.140s	35.67s	–	0.148s
PMOO	0.035s	17.67s	–	0.035s

Delay bounds linear: GPC $\approx$ 72, PBOO $\approx$ 57, PMOO $\approx$ 23

Delay bounds others: GPC = 21, PBOO = 19, PMOO = 15

Running example from [Guan,Wang'13] Prefix sizes

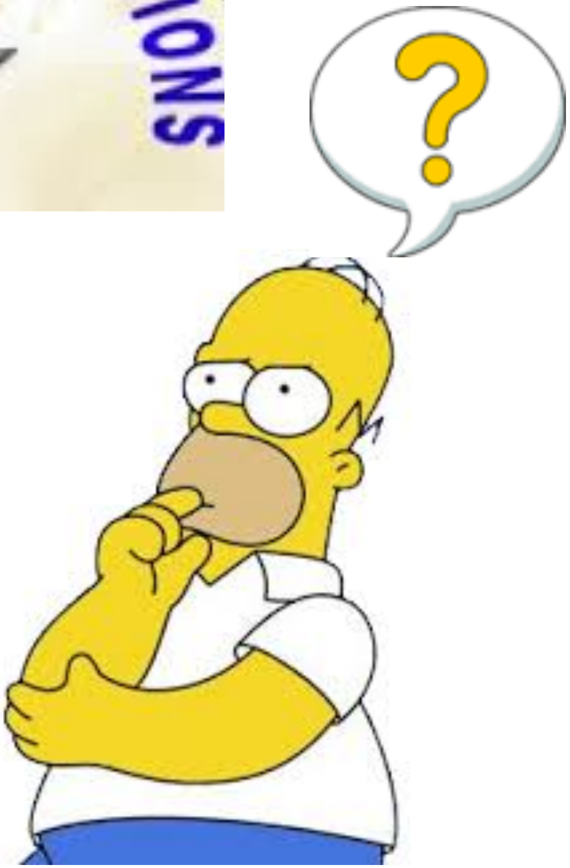


(b) Component-wise prefixes for the GPC analysis

	R_1		R_2		R_3	
	C_i-finit	$\odot$-finit	C_i-finit	$\odot$-finit	C_i-finit	$\odot$-finit
E_1	98	81	90	61	74	35
E_2	90	81	78	61	59	35
E_3	78	81	60	61	35	35
E_4	60	56	35	35	35	35

Conclusion

- ✧ **For large systems, the representation of the arrival and services curves gets complex very fast as experienced with the HCS model**
- ✧ **Function prefixing avoids this by limiting curves to finite domains, HCS case study provided evidence but lacked formal criterion on the prefix size.**
- ✧ **Finitary RTC provided sucha criterion, but is limited to GPC-models and their flow equations.**
- ✧ **but, GPC-based system modelling is unaccpetable, runtime and precision-wise.**
- ✧ **This called for re-visiting of function prefixing, but at the level of individual RTC-operators**



Thank you for your time