# Reliable communication and control
## for the Smart Grid

Utkarsh Upadhyay

**Supervisor:** Prof. Jean-Yves Le Boudec
**Advisor:** Dan Cristian Tomozei
**Lab:** LCA2

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

February 14, 2012

# Outline

Introduction

Problem

Verifying the contract

Predicting guaranteed signal

Implementation

Demo

Assessment

Future work

# The world today

- Energy producers are moving from traditional sources
- Renewables have high variability

## Managing the difference

- Storage at source
- Fossil fuel backups
- Demand response
- ...

Smart grids enable Demand response

# The world today

- Energy producers are moving from traditional sources
- Renewables have high variability

## Managing the difference

- Storage at source
- Fossil fuel backups
- Demand response
- ...

Smart grids enable Demand response

# The world today

- Energy producers are moving from traditional sources
- Renewables have high variability

## Managing the difference

- Storage at source
- Fossil fuel backups
- Demand response
- ...

Smart grids enable Demand response

# The Demand-Response problem





Demand-Response is an mechanism to manage customer consumption of electricity (demand) in response to supply conditions.

Current approaches:

1. Dynamic Pricing
   - Interesting but has problems[a]
2. Delayed supply
   - Promising but currently ad-hoc; deployed by Votalis, Peaksaver

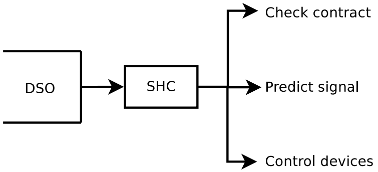
Le Boudec *et. al.* formalize 2 via service curve contracts.

[a] May expose consumers to price volatility

# Outline

# Definition

## Objectives




## Intermediate goals

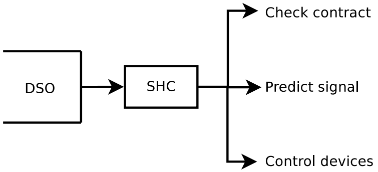
- A Communication protocol.
- Smart Home Controller (SHC) architecture.

## Design constraints

- Reliability and robustness.
- Computational feasibility.

# Definition

## Objectives




## Intermediate goals

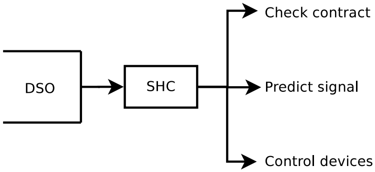
- A Communication protocol.
- Smart Home Controller (SHC) architecture.

## Design constraints

- Reliability and robustness.
- Computational feasibility.

# Definition

## Objectives




## Intermediate goals

- A Communication protocol.
- Smart Home Controller (SHC) architecture.

## Design constraints

- Reliability and robustness.
- Computational feasibility.

# Tools of trade

- Machine running Linux
- Capable of communication
- UI/Algo module can be distributed

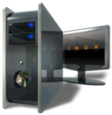


Fig: ZPlug from Cleode

- ZigBee SE profile compliant
- Passive monitoring devices
- Can turn devices ON/OFF

# Outline

# Service curve $\beta(\cdot)$

- Borrowed from Network Calculus
- Minimum energy in time $t = \beta(t)$.
- Usually made by repeating the *derivative* with period $t'$
- Are deterministic.
- Slight abuse of notation on slides.

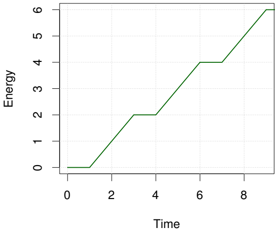



Fig: Serv. cur. (period = 3)

## What makes it feasible

Keeping just one period $t'$ of $u(\cdot)$ is sufficient to verify the service curve constraint, for binary service curves and bounded $u(\cdot)$.

# Service curve $\beta(\cdot)$

- Borrowed from Network Calculus
- Minimum energy in time $t = \beta(t)$.
- Usually made by repeating the *derivative* with period $t'$
- Are deterministic.
- Slight abuse of notation on slides.

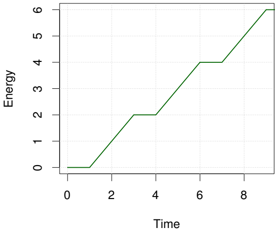



Fig: Serv. cur. (period = 3)

## What makes it feasible

Keeping just one period $t'$ of $u(\cdot)$ is sufficient to verify the service curve constraint, for binary service curves and bounded $u(\cdot)$.

# Service curves constraint

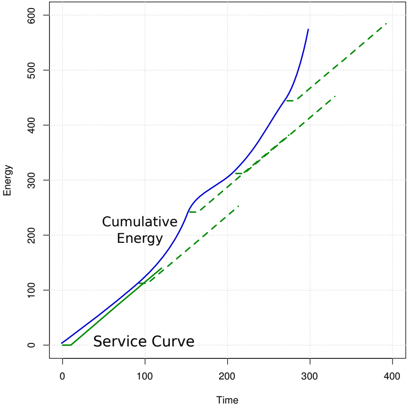



Fig: Control signal constrained by the service curve

$$\forall t_1, t_2.\ \int_{t_1}^{t_2} u(\tau)\, d\tau \geq \beta(t_2 - t_1) \implies U(t_2) - U(t_1) \geq \beta(t_2 - t_1)$$

# Service curve violation

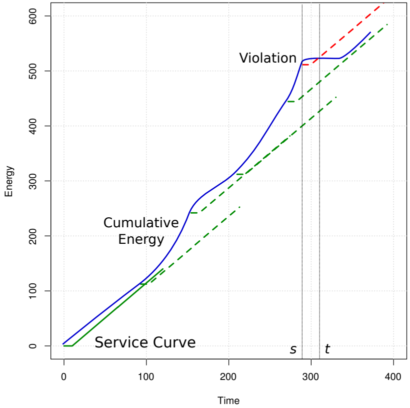


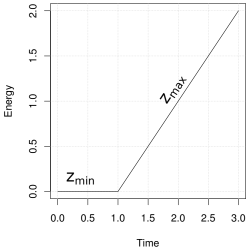

Fig: Service curve violation

$$\exists t, s.\ U(t) - U(s) \leq \beta(t - s)$$

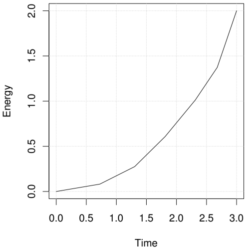
$s$, $t$ are not unique

# Generalized the result

**Theorem:** Keeping just one period of $u(\cdot)$ is sufficient to verify service curve constraint.




Binary serv. cur.




Star shaped serv. cur.

## Constr. dropped

- $u(\cdot)$ was bdd above by $z_{max}$
- $\beta(\cdot)$ was a binary serv. cur.

## Shortcomings

- Computationally more complex

# Outline

# Minimum guaranteed signal

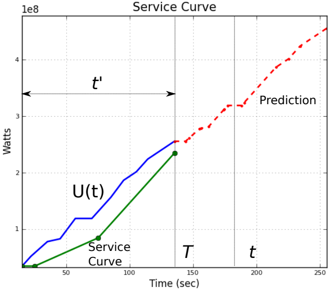



Fig: Minimum signal guaranteed

$$\hat{U}(t) \geq \sup_{s \in [0, T]} \{U(s) + \beta(t - s)\}$$

# Minimum guaranteed signal

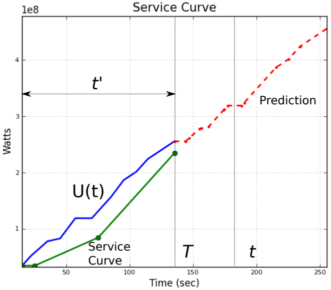



Fig: Minimum signal guaranteed

$$\hat{U}(t) \geq \sup_{s \in [t-t', T]} \{U(s) + \beta(t-s)\}$$

- Naive algorithm (discrete Max-convolution)[b] $\approx O(S^2)$
- Quota allocation policy algorithm (Binary service curve): $\approx O(S)$

[b] Complexity per prediction calculation, $S$ is number of samples in one period

# Prediction algorithm

## Issues with naive algorithm

- Worked in discrete time, limited accuracy
- Was computationally expensive (5 sec. for 24 hours ≈ 300 million ops./calc.)
- Efficient version was limited to binary service curves

## New algorithm

- Works in continuous time, has more accuracy.
- Has complexity[c] $O(N \cdot K)$
  - More efficient than naive version even if control signal changes each sampling time
  - Is equal to complexity of efficient algorithm for binary curves
- Non-trivial, but proved correct.
- Complexity depends on input

## Design recommendation

- Control $N$ and $K$ while forming the contract.

[c] $K = \#$ changes in $u(\cdot)$ in the last period, $N =$ segments in Service Curve

# Prediction algorithm

## Issues with naive algorithm

- Worked in discrete time, limited accuracy
- Was computationally expensive (5 sec. for 24 hours $\approx$ 300 million ops./calc.)
- Efficient version was limited to binary service curves

## New algorithm

- Works in continuous time, has more accuracy.
- Has complexity[c] $O(N \cdot K)$
  - More efficient than naive version even if control signal changes each sampling time
  - Is equal to complexity of efficient algorithm for binary curves
- Non-trivial, but proved correct.
- Complexity depends on input

## Design recommendation

- Control $N$ and $K$ while forming the contract.

[c] $K = \#$ changes in $u(\cdot)$ in the last period, $N =$ segments in Service Curve

# Prediction algorithm

## Issues with naive algorithm

- Worked in discrete time, limited accuracy
- Was computationally expensive (5 sec. for 24 hours $\approx$ 300 million ops./calc.)
- Efficient version was limited to binary service curves

## New algorithm

- Works in continuous time, has more accuracy.
- Has complexity[c] $O(N \cdot K)$
  - More efficient than naive version even if control signal changes each sampling time
  - Is equal to complexity of efficient algorithm for binary curves
- Non-trivial, but proved correct.
- Complexity depends on input

## Design recommendation

- Control $N$ and $K$ while forming the contract.

[c] $K = \#$ changes in $u(\cdot)$ in the last period, $N =$ segments in Service Curve

# Prediction algorithm

## Issues with naive algorithm

- Worked in discrete time, limited accuracy
- Was computationally expensive (5 sec. for 24 hours $\approx$ 300 million ops./calc.)
- Efficient version was limited to binary service curves

## New algorithm

- Works in continuous time, has more accuracy.
- Has complexity[c] $O(N \cdot K)$
  - More efficient than naive version even if control signal changes each sampling time
  - Is equal to complexity of efficient algorithm for binary curves
- Non-trivial, but proved correct.
- Complexity depends on input

## Design recommendation

- Control $N$ and $K$ while forming the contract.

---

[c] $K = \#$ changes in $u(\cdot)$ in the last period, $N =$ segments in Service Curve

# Outline

# Smart Home Controller

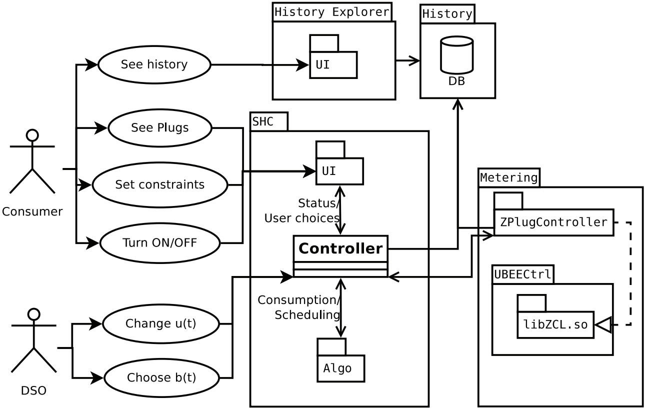



Fig: The Architecture

# Smart Home Controller

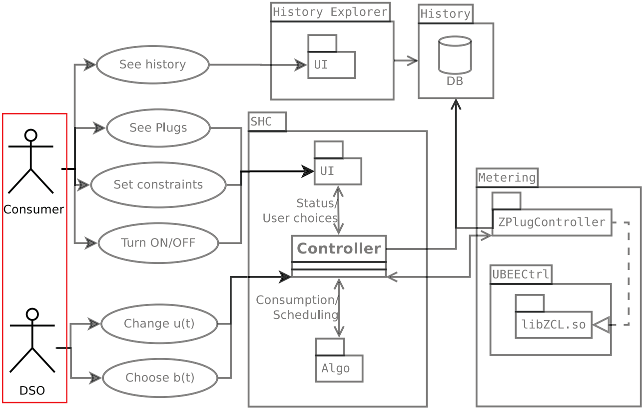



Fig: Actors

# Smart Home Controller

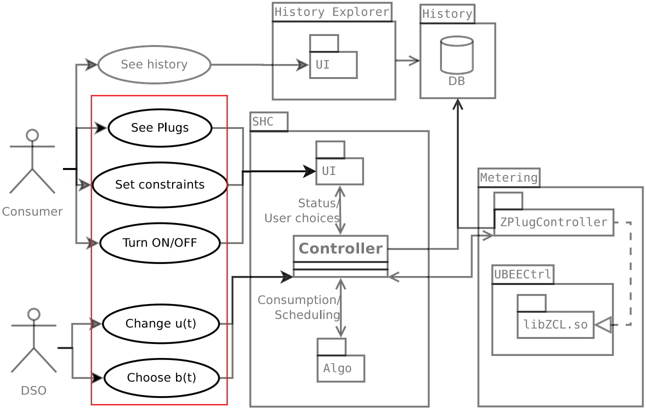



Fig: Actions

# Smart Home Controller

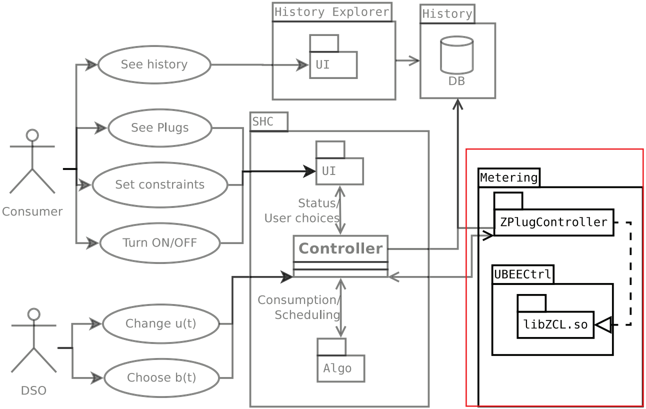



Fig: ZPlugs

# Smart Home Controller

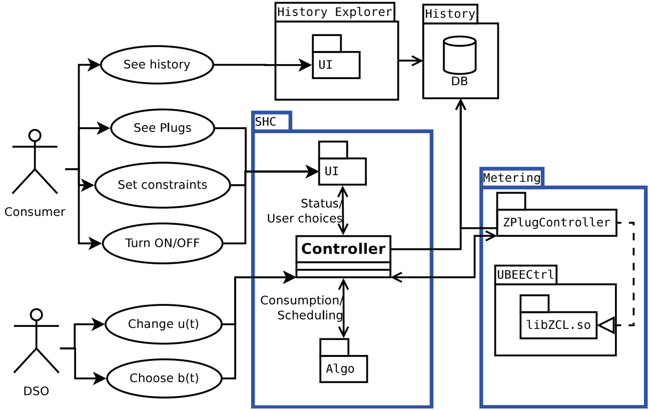



Fig: SHC architecture

# Communication protocol

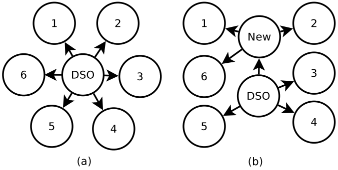



Fig: Messaging layer can manage overloading and grouping

- Communication happens via discrete messages
  - *no continuous control*
  - Data representation scheme is designed
- PUB-SUB model is employed instead of REQ-REP.

# Communication protocol

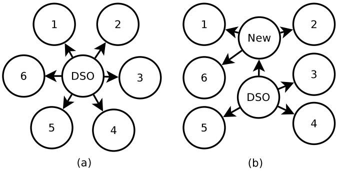



Fig: Messaging layer can manage overloading and grouping

## Pub-Sub v/s Req-Rep

- Scalability
- Separation of concerns
  - Grouping subscribers
- Logging & Diagnostics
- Simplicity

## Problem

- Newly started SHC
- Periodic broadcast

# Communication protocol

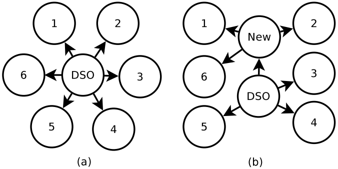



Fig: Messaging layer can manage overloading and grouping

## Pub-Sub v/s Req-Rep

- Scalability
- Separation of concerns
  - Grouping subscribers
- Logging & Diagnostics
- Simplicity

## Problem

- Newly started SHC
- Periodic broadcast

# Communication protocol

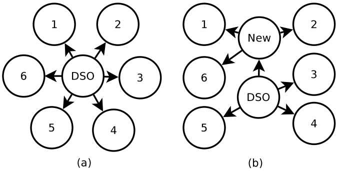



Fig: Messaging layer can manage overloading and grouping

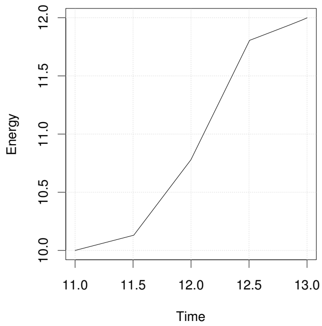
## Pub-Sub v/s Req-Rep

- Scalability
- Separation of concerns
  - Grouping subscribers
- Logging & Diagnostics
- Simplicity

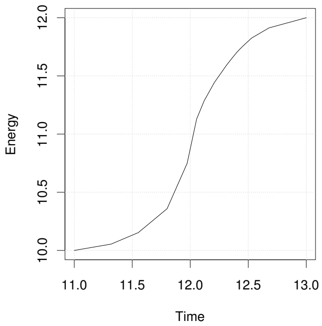
## Problem

- Newly started SHC
- Periodic broadcast

# Discrete v/s continuous time




Discrete time




Continuous time

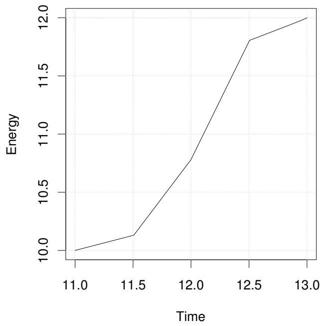
## Discrete

- Ease of presentation
- Sampling period trade-off:
  - Accuracy
  - Scalability

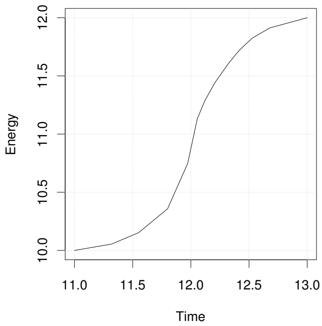
## Continuous

- Better accuracy
- One less parameter
- Makes prediction trickier

# Discrete v/s continuous time




Discrete time




Continuous time

## Discrete

- Ease of presentation
- Sampling period trade-off:
  - Accuracy
  - Scalability

## Continuous

- Better accuracy
- One less parameter
- Makes prediction trickier

# Data representation

## Control signals

- Discrete signals $\implies$ inform only of changes
- Cumulative energy is piecewise linear

## Service Curves

- As a piecewise linear function
- Can approximate any function
- Usually just binary

## Piecewise linear continuous functions

- Store end points of all segments.

# Data representation

## Control signals

- Discrete signals $\implies$ inform only of changes
- Cumulative energy is piecewise linear

## Service Curves

- As a piecewise linear function
- Can approximate any function
- Usually just binary

## Piecewise linear continuous functions

- Store end points of all segments.

# Data representation

## Control signals

- Discrete signals $\implies$ inform only of changes
- Cumulative energy is piecewise linear

## Service Curves

- As a piecewise linear function
- Can approximate any function
- Usually just binary

## Piecewise linear continuous functions

- Store end points of all segments.

# Reliability

## Primary problems

- Crashes: Application or a dependency might crash
- Live-locks: Operations might start taking too long or freeze
- Byzantine faults: Nonsensical output (hopefully)

# Reliability

## Primary problems

- Crashes: Application or a dependency might crash
- Live-locks: Operations might start taking too long or freeze
- Byzantine faults: Nonsensical output (hopefully)

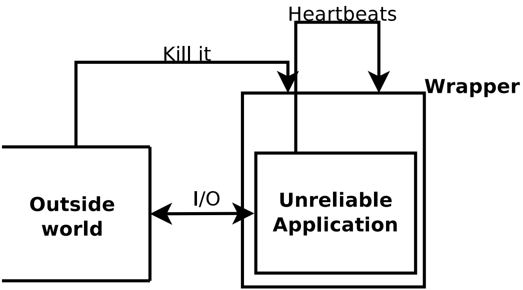



Fig: Wrapper design

Hosted at: `https://github.com/musically-ut/pyoolproof`

# Reliability

## Primary problems

- Crashes: Check for error codes, restart application, persist data
- Live-locks: Listen to heartbeats, kill and restart if too slow
- Byzantine faults: Kill on request (STONITH)

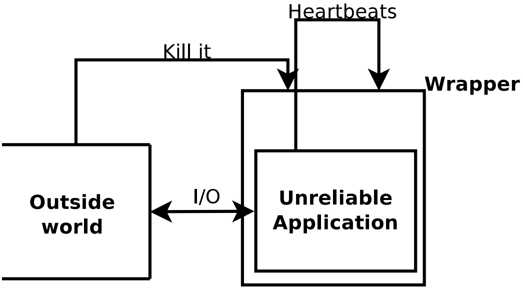



Fig: Wrapper design

Hosted at: `https://github.com/musically-ut/pyoolproof`

# Implementation

| Language | Lines of code | Comments | Component |
|---|---|---|---|
| Python | 4,300 | 1,300 | UI, Test harness |
| C++ | 800 | 220 | ZPlug wrapper |
| Ocaml | 2,000 | 460 | SHC |
| **Total** | 7,100 | 2,080 | – |

Tbl: Details about the code in the implementation

- Battery of unit tests
- Good documentation and examples

# Outline

# Outline

# Skills acquired

## Demand-response using Service Curves

- Network Calculus fundamentals
- Demand response

## Robust software design

- **Ocaml:** Functional + OO paradigm
- **Modular design:** Separating points of failures

## Tools

- **ZeroMQ:** TCP sockets on steroids
- **SQLite:** Embedded reliable persistence.
- Also learned about ZigBee and ZPlugs.

# Assessment

## State of testbed

- Currently fully functional
- Easy to extend

## Self assessment

- Stress testing
- Deployment
- Theoretical results were time consuming

# Outline

# Future work

## Testbed

- Systematic stress testing.
- Formal verification.

## Prediction and Optimization

- Making the algorithm incremental

## Design

- Security

# Thank you

Utkarsh Upadhyay
mail@utkarshu.in

# Outline

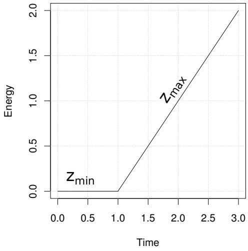
Appendix

# Prediction algorithm

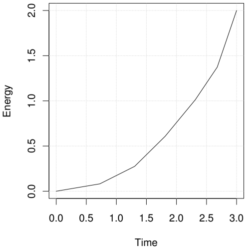
**Input:** $\beta(\cdot) = [(s_1, \beta(s_1)), \ldots, (s_N, \beta(s_N))]$,
$U(\cdot) = [(t_1, U(t_1)), \ldots, (t_K, U(t_K))]$
**Output:** predicted segments/function

$\hat{V}_1 = [(t_K + s_i, U(t_K) + \beta(s_i))$ for $s_i$ in $\mathbf{T}_\beta]$
scTip $= (t_K,\ \beta(s_N) + U(t_K))$
**for** $j = K - 1$ to $1$ **do**
    $t_j = \mathbf{T}_U[j]$
    C $= [(t_j + s_i, U(t) + \beta(s_i))$ for $s_i$ in $\mathbf{T}_\beta]$
    C.*add(scTip)*
    $\hat{V}_{(K-j+1)} =$ maxCover($\hat{V}_{(K-j)}$, C)
    $\hat{V}_{(K-j+1)}$.*limitTime*($t_K,\ t_K + s_N$)
    scTip $= (t_j + s_N,\ U(t_j) + \beta(s_N))$
**end for**
**return** $\hat{V}_K(\cdot)$

# Generalization




Binary serv. cur.




Star shaped serv. cur.

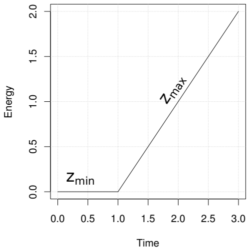
$u(t)$ is control signal defined up to some time horizon $T$.

**Theorem:** *If $u(t) \leq z_{max}$ for all $t \leq T$, then these are equivalent:*

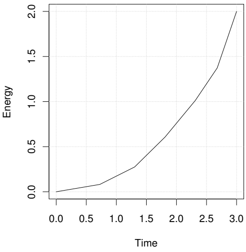
1. $\int_s^t u(\tau)\, d\tau \geq \beta(t - s)$ *for all* $s < t \leq T$,
2. $\forall \tau \leq T.\, u(\tau) \geq z_{min}$ *and* $\int_t^{t+t'} u(\tau)\, d\tau \geq \beta(t')$, $\forall\, t$ *such that* $t + t' \leq T$.

*where $\beta(\cdot)$ is a binary service curve.*

# Generalization




Binary serv. cur.




Star shaped serv. cur.

$u(t)$ is control signal defined up to some time horizon $T$.

**Theorem:** *If control signal $u(\cdot)$ violates service curve $\beta(\cdot)$ for the first time at time $T$, then*

$$\exists\, t \in [T - t', T).\, U(t) + \beta(T - t) > U(T)$$

# Generalization

**Previous result:** *If $u(t) \leq z_{max}$ for all $t \leq T$, then these are equivalent:*

1. $\int_s^t u(\tau)\, d\tau \geq \beta(t-s)$ *for all* $s < t \leq T$,
2. $\forall \tau \leq T.\, u(\tau) \geq z_{min}$ *and* $\int_t^{t+t'} u(\tau)\, d\tau \geq \beta(t')$, $\forall\, t$ *such that* $t + t' \leq T$.

*where $\beta(\cdot)$ is a binary service curve.*
**New result**: *If control signal $u(\cdot)$ violates service curve $\beta(\cdot)$ for the first time at time $T$, then:*

$$\exists\, t \in [T - t', T).\, U(t) + \beta(T - t) > U(T)$$

## Constr. dropped

- $u(\cdot)$ was bdd above by $z_{max}$
- $\beta(\cdot)$ was binary serv. cur.

## Restrictions

- Max-plus convolution