An adaptive systems perspective on network calculus, with applications to autonomic control

Simon Dobson

Systems Research Group, UCD Dublin IE

simon.dobson@ucd.ie

Introduction

Adaptive systems, and especially *communications* systems, pose significant challenges for designers

- Multiple small-scale behaviours
- . . . co-ordinated to maintain a single overall behaviour
- . . . with minimal human intervention

What can we do to improve the way we design such systems? How can we be sure our solutions will work, and *keep* working?

What we will and won't have

This talk will have:

- An identification of the main challenges
- An outline of a possible approach to analysis and design
- Some pointers to making it all work

What we will and won't have

This talk will have:

- An identification of the main challenges
- An outline of a possible approach to analysis and design
- Some pointers to making it all work

This talk won't have:

- Heavy maths being worked out in front of you
- Much in the way of concrete results

Where I'm from

Ireland

- An island off the north-west coast of France, famous for its rain, potatoes and alcohol addiction

UCD Dublin

- Largest university in Ireland – 20,000 undergrads and 5,000 grad students

Systems Research Group (SRG)

- Adaptive and pervasive systems, languages and middleware, dependable software engineering, visualisation, low-power systems and sensor design

Why autonomics? – 1

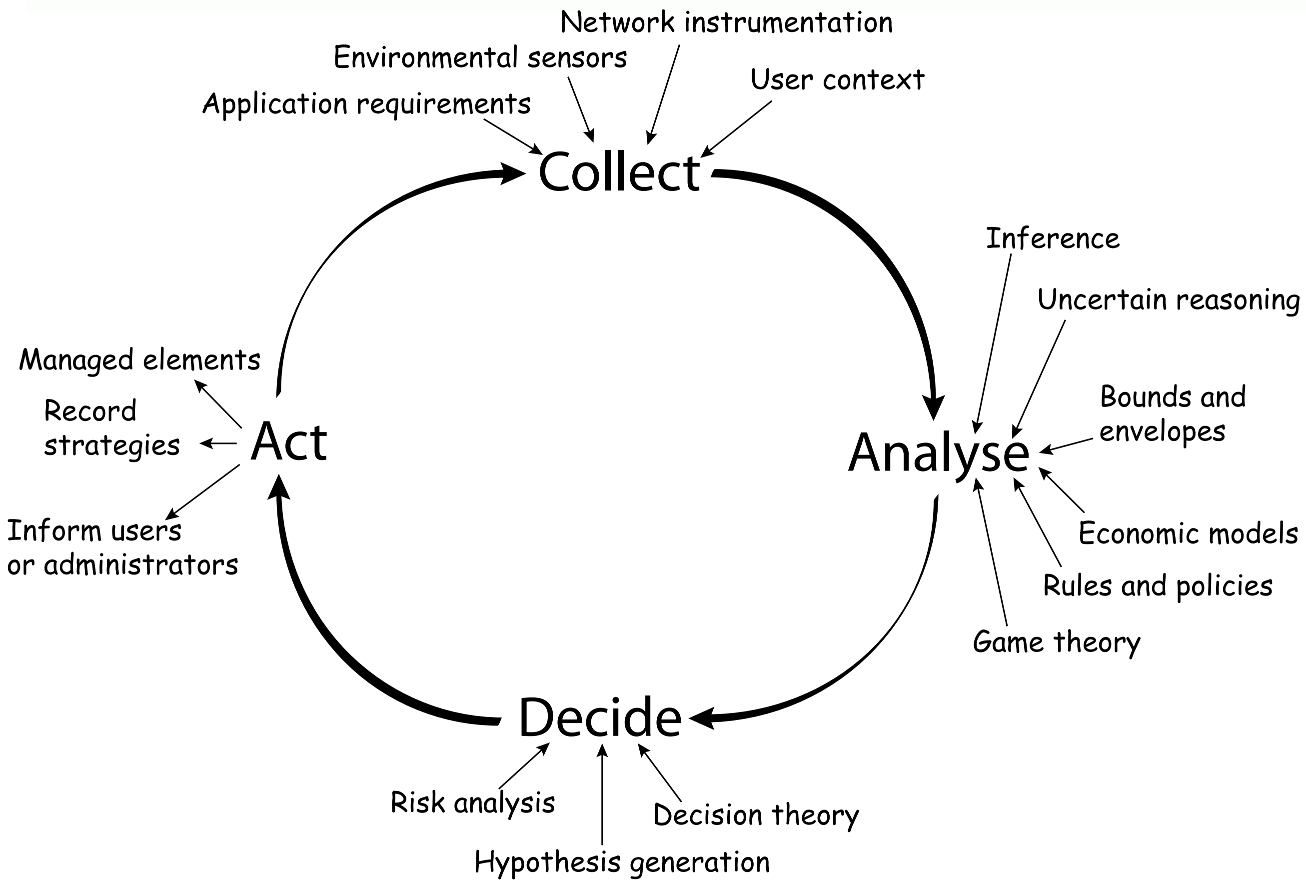
Complexity of modern systems

- Difficult to make changes, even as the customer experience has (often) become easier
- Can't support agile business, narrow-window opportunities
- Doesn't support IT as a profit centre

Use more technology in the management of technology

- "Close the loop" on control
- Let a management system observe, and react to, changes in behaviour and envronment

Why autonomics? – 2



(From Dobson et alia. A survey of autonomic communcations. ACM Trans. Autonomous and Adaptive Systems ***1****(1). December 2006.)*

Approaches

Two general approaches

1. Take a traditionally-engineered system, add sensing and reasoning to affect the available control levers (power management, sevrer provisioning)
2. Find approaches that are inherently stable under perturbation (routing, data dissemination)

The former is less intrusive, but perhaps harder to scale; the latter can be more effective, but means re-building systems *ab initio*

Desiderata

We need our systems to be:

Desiderata

We need our systems to be:

- *Comprehensible*, to generate confidence through analysis and simulation

Desiderata

We need our systems to be:

- *Comprehensible*, to generate confidence through analysis and simulation
- *Compositional*, to allow proper engineering and evolution

Desiderata

We need our systems to be:

- *Comprehensible*, to generate confidence through analysis and simulation
- *Compositional*, to allow proper engineering and evolution
- *Open*, to allow exploration and innovation

Desiderata

We need our systems to be:

- *Comprehensible*, to generate confidence through analysis and simulation
- *Compositional*, to allow proper engineering and evolution
- *Open*, to allow exploration and innovation
- *Verifiable*, so we can convince people they got what they paid for

Getting these properties

We claim that, to get these properties, we need a combination of:

Formal analysis Prove the properties we want. Relate adaptive behaviour from descriptions of the stimuli to which the system should adapt

Structured design Derive code from description. Have systems correct by construction, rather than try to prove correctness *post facto*

Summary of what's to come

We've been looking at how we might treat systems adaptation as a problem of *mapping* and of *dynamical systens*

- *Mapping*: describe the context of a system and its acceptable behvioural variations, map one to the other
- *Dynamics*: different strategies are simply different navigations of the behavioural space

In this talk we apply this general approach to the specific instance of network modeling using network calculus

Network calculus

Network calculus is a relatively new formalism that's starting to gain traction as a way of doing analytic modelling on complex networks

Le Boudec and Thiran. Network calculus: a theory of deterministic queueing systems for the internet. LNCS 2050. 2001.

- Model network elements by their impact on flows
- Constrain arrival and service curves, as done in IntServ
- Analyse performance, virtual delays, throughput, . . .

Basic idea

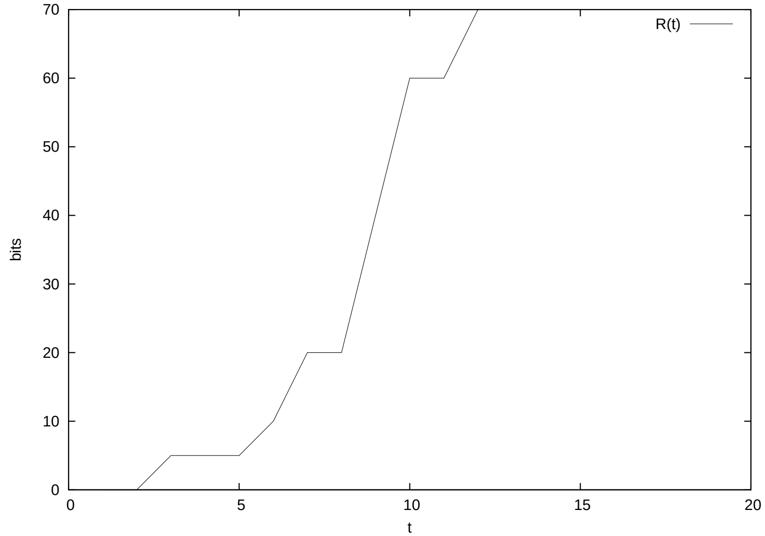
Min-plus algebra

- Normal control theory works on normal algebra, a dioid $(\mathcal{R}, +, \times)$
- Network calculus uses an alternative dioid $(\mathcal{R}, \inf, +)$ – so "addition" is taking a minimum and "multiplication" is addition

(Why? Because this lets us always constrain the *maximum* number of bits moving in a part of the network at any time)

Flows and curves – 1

An *input function* $R(t)$ models the number of bits that have arrived at an element at time t



- The flow function is cumulative (*wide-sense increasing*)

- Can model using discrete or continuous time – for simplicity we'll stick to continuous, although it's an approximation of how real elements behave

Flows and curves – 2

Similarly, an *output function* $R^*(t)$ captures the total number of bits that have flowed from an element at time t

- Also wide-sense increasing

Constrain $R^*(t) \leq R(t)$ for all t: what goes out must have come in

- Or, to put it another way, elements *process* traffic they don't *create* it

By convention $R(t) = R^*(t) = 0$ for all $t \leq 0$

Arrival and service curves – 1

An input function tells us *what* traffic arrives, but we also need to know *how* it arrives

- Some traffic is smooth, other traffic is “bursty”
- Typical multimedia traffic has isochrony requirements as well as bandwidth requirements

An *arrival curve* $\alpha(t)$ defines traffic’s shape

- $R(t)$ *conforms* to $\alpha(t)$ iff $R(t) - R(s) \leq \alpha(t - s)$ for all $s \leq t$
- α constains the volume of traffic that can arrive in any given time interval

Arrival and service curves – 2

Similarly, a *service curve* $\beta(t)$ constrains how traffic leaves an element

- $R^*(t) - R(s) \geq \beta(t - s)$ for some $s \leq t$

These curves model the core behaviour of elements

- Arrival curves model the worst traffic patterns an element is *expected* to be able to deal with
- Service curves model the traffic the element *guarantees* to serve out at least
- You can see a service curve as a *reservation*: it specifies the trffic shape we expect to produce

Leaky buckets and lagged pipes

The canonical arrival curve is the *leaky bucket*

- Maintain a traffic flow of rt over the long term, but allow "bursts" of up to b bits per second
- $\gamma_{r,b} = rt + b$

The classic service curve is the *rate latency* curve

- Serve traffic at a rate R, lagging behind the input with a latency of T for processing time within the element
- $\beta_{R,T} = R.max(t - T, 0)$

(De-)convolution

Given these basic descriptions, we need to combine flows in a way that preserves their significant properties

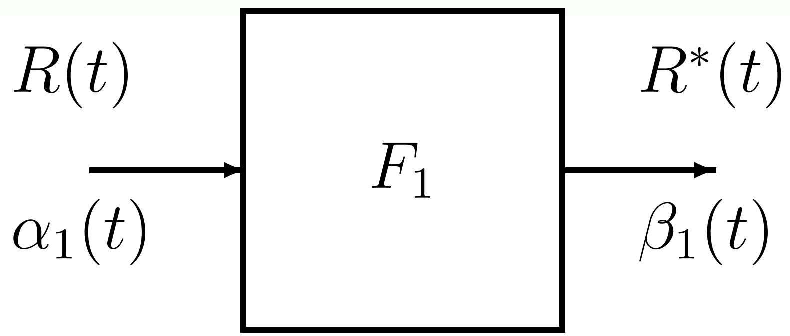
Convolution takes one flow and convolves it with another

- The "degree of overlap" between the two
- $(f \otimes g)(t) = \inf_{0 \leq s \leq t} [f(t-s) + g(s)]$

Dually, *deconvolution*

- $(f \oslash g)(t) = \sup_{s \geq 0} [f(t+s) + g(s)]$

Example – 1

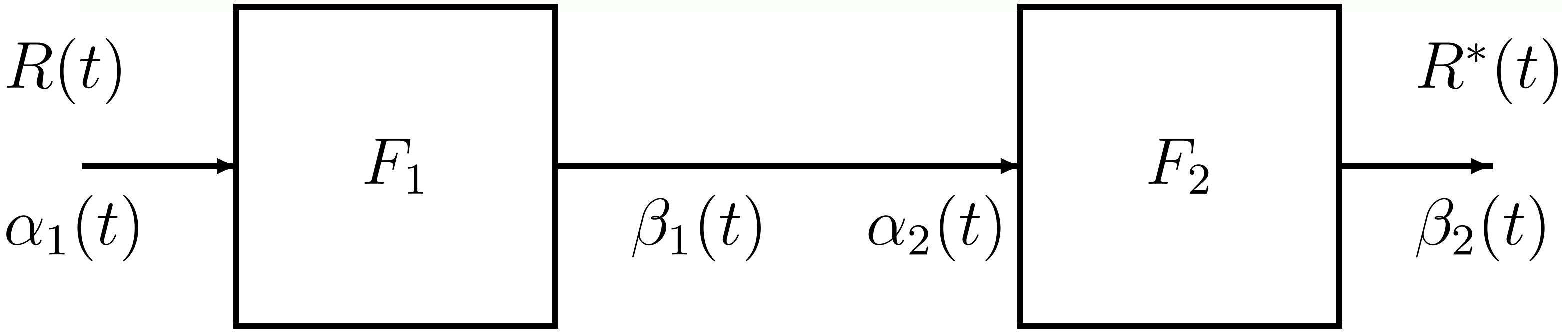


The *backlog* of data in this system satisfies

$$R(t) - R^*(t) \leq \sup_{s \geq 0}[\alpha_1(s) - \beta_1(s)]$$

- The traffic remaining to be served at time t
- The largest difference allowed by the arrival and service descriptions

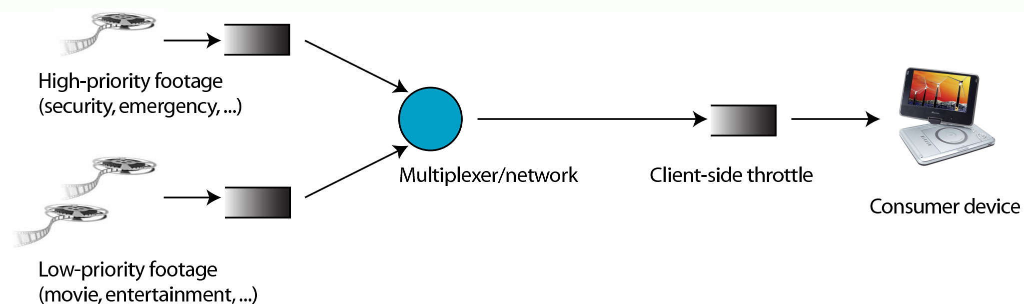
Example – 2

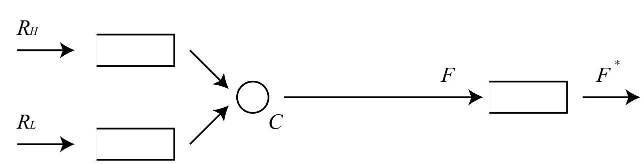


Moreover the output $R^*(t)$, as well as guaranteeing service β_1, also conforms to an arrival curve $\alpha_1 \oslash \beta_1$
If we add another element with service curve β_2, the combined system will offer a service curve $\beta_1 \otimes \beta_2$

- The first service curve constrains the second
- For rate latency curves,
  $\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2}$

A systems perspective – 1





- A high-priority flow R_H and a low-priority flow R_L
- A non-pre-emptive multiplexer delivering a constant rate T, preferring traffic from R_H
- A traffic shaper for the end-point

A systems perspective – 2

It can be show (after some tricky calculation. . .) that:

- If R_L has a packet size l^L_{max} and R_H arrives at a rate $r < C$, then
- High-priority traffic is served according to $\beta_{C,\frac{l^L_{max}}{C}}$
- Low-priority traffic is served according to $\beta_{C-r,\frac{b}{C-r}}$ when R_H is quiescent

This sounds ridiculously abstract – but it isn't

- Very like the parameters used in IntServ and DiffServ
- Implementation can retain confidence in the analysis

An adaptive *systems perspective*

So far we can state properties of a network and perform calculations about it

- Throughput, traffic shaping, . . .

For an *adaptive* system we also want to study how the system's behaviour changes with its changing environment

- Select control actions based on changing, sensed environment
- External constraints, network behaviour, user intensions, . . .

Power-aware networking

Wireless networks, and especially wireless *sensor* networks, increasing seek to be power-aware

- Increase node lifetime, improve focus on significant events

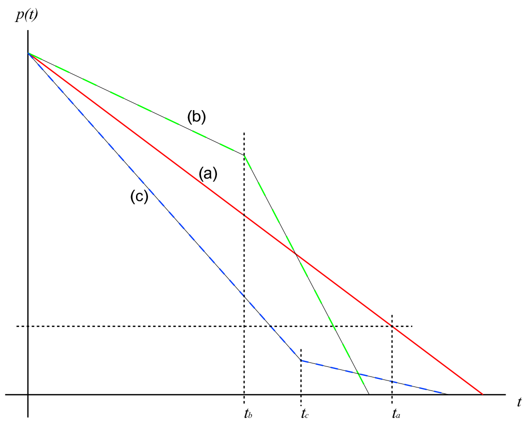
Rule-based approaches can work in simple cases, but may be a little *too* simple

- Tie the sensed context directly to (a suite of possible) control actions

How can we model changing behaviour in a principled way?

Bandwidth reduction

One possible response to reducing power is to throttle bandwidth



- $p(t) < p(t_a) \Rightarrow$ reduce U, the endpoint bandwidth
- Can't simply reduce capacity, as buffer must stay finite
- Reduce multiplexer rate C to remain below U
- ...which forces $U \geq r$ to handle R_H's traffic
- ...so reduce r, or lose packets

What we're doing

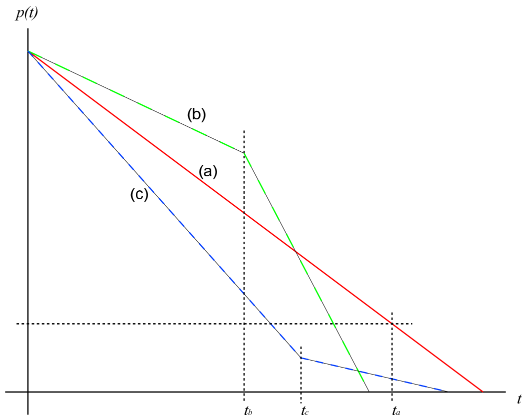
The point here is not the specific control actions we might take, but our ability to model them (and their effects) precisely

Put another way, control actions form a function from p to the control parameters we have access to

- Can't decide entirely on the most desirable action just from power
- Is frame loss acceptable? Is degradation preferable? Should we (or can we) drop some flows entirely?
- Set out the space, use other information to make these decisions

Other actions

However, the space of models is larger than this suggests



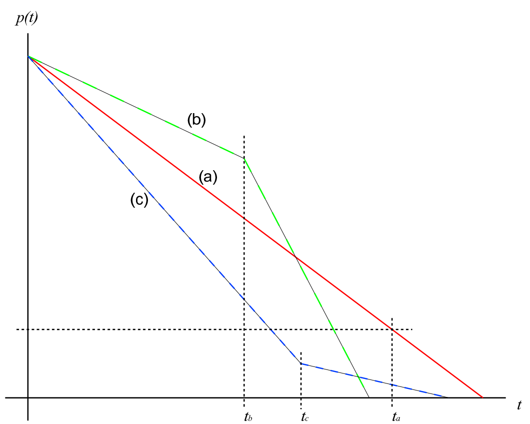
- A steeply-falling power reserve may need different actions to a more stable one
- Depend on $\frac{dp}{dt}$
- Earlier intervention may open-up alternatives

Look for "smooth" designs

- Small changes in context lead to small control "nudges"
- Not always possible, *i.e.* emergency shutdown
- Tolerate errors in sensing

Hypothesis testing

Because we have a closed loop, we can treat control actions as hypotheses that we then test against reality



- If we intervene at t_c, we expect to see a reduction in power depletion as a result: $\frac{dp}{dt}(p + \Delta t) < \frac{dp}{dt}(t_c)$
- …and if we don't see this, we can try another action

Compositionality

We can take this argument a stage further and allow control parameters to be composed

- Depend on $\frac{\partial p}{\partial t}$ as we add more dimensions that vary across the system's lifetime

The portfolio of control actions can simlarly be enriched

- Include discrete actions, plans (in the AI sense), . . .
- Non-standard sensing such the "meaning" of a flow to a user

Pose and solve these problems dynamically

- Different to classical control theory: less precise and predictive, more symbolic and dynamic

The dynamic control space

An alternative view is that we're forming a state space for the system which we then navigate using out portfolio of control actions

- The dimensions of the space, the actions available, and the co-dimension of effects may all change dynamically

We conjecture that we can state the "envelope of behaviour" the system stays in, and describe navigation strategies

- ... although this is *only* a conjecture for now...

Conclusion

We have tried to present a sketch of an approach to autonomic systems design that harnesses – but to an extent moves beyond – conventional modeling and control theory

- Model networks using an emerging, accepted formalism
- Select possible control actions based on changes in observed context
- Verify hypotheses and try out alternative strategies

Our small-scale demonstrations now need to be scaled up and rigourously tested, both analytically and in practice

5 points to take away

5 points to take away

- Autonomic systems need to be comprehensible, compositional, open and verifiable

5 points to take away

- Autonomic systems need to be comprehensible, compositional, open and verifiable
- We can learn from control theory and network modeling, and relate observation to control in a structured way

5 points to take away

- Autonomic systems need to be comprehensible, compositional, open and verifiable
- We can learn from control theory and network modeling, and relate observation to control in a structured way
- We have a richer space to play in because we have non-standard actions, plans and information sources

5 points to take away

- Autonomic systems need to be comprehensible, compositional, open and verifiable
- We can learn from control theory and network modeling, and relate observation to control in a structured way
- We have a richer space to play in because we have non-standard actions, plans and information sources
- Our control systems form dynamic explorations of these spaces, for which there is a rich mathematical underpinning

5 points to take away

- Autonomic systems need to be comprehensible, compositional, open and verifiable
- We can learn from control theory and network modeling, and relate observation to control in a structured way
- We have a richer space to play in because we have non-standard actions, plans and information sources
- Our control systems form dynamic explorations of these spaces, for which there is a rich mathematical underpinning
- Models of behavioural spaces may allow us to make guarantees that can be relied upon