Lecture Notes

Analytical Network & System Admin.

1) PHILOSOPHY OF SCIENCE CHAP 1,2

How to understand system administration

Sysadmin:
- planning
- deployment
- maintenance

Tech systems:
- design (abstract)
- reliability (evaluation)
- efficiency / integrity
- success criteria? (eval.).

Not just technology! Users interact!

Human-computer systems
- unreliable / unpredictable
- policy goals - what's it for?
- security (feeling safe).

So - how can we build a theory?
- Who/what are the players?
- Variables?
- Measurables?
- What interactions take place?
- Model these to make predictions.

e.g. data centre design:
- heat, power, work rate, arrivals, completions, transactions.

Modelling:

E~~mpiricism~~ (Francis Bacon) - look for characteristics / exceptions to rule

Descartes - inspired by Geometry (crystal) ⇒ Newton

John Locke - inspired by Newton - calculate the future! define 'empiricism'

David Hume - 2 kinds of knowledge
- uncertain knowledge about real world
- certain knowledge about theoretical world

I. Kant - perceptions play a role.

K. Popper - falsification. Can only prove false hypothesis.

~~Feyerbend~~

Technology - introduces a subjective purpose to the world.

⇒ value judgements (suitability).

Today these philosophies have split into two main branches:
science ⇒ empiricism (conventionally)

Empiricism: observe, form hypothesis to try to explain.

Theory / math: assume, then try to deduce consequences.

"

Comments
- No theory / observation is perfect "true". We are therefore looking for suitably idealized approximations
  reasonable explanation.
- Law of causality. Cause precedes effect.

How good is the scientific method?
- encourage reliable answers to searching questions.
- Get it wrong? Ethical duty?

J.S. Mill - science is self-correcting.

Abuses of science:
- cheating, manipulation, forgery
- seeing what you want to see

Non-seq: "scientifically proven".

Marketing: "ologies"

e.g. positions of exploding gas clouds 100 billion years ago tells us if we will fall in love?

Arrogance

- power lines / mobile phones
- people said it was impossible for a bumble bee to fly.
- Martian canals.

> We cannot be certain we are right, only that we ~~have been~~ are being sufficiently critical.

Scales of Measurement

How many are you?
How much is the ceiling?

What is the difference between a kilometre and a kilogramme?

- arbitrary scales
- initially unrelated concepts, give different names
- "engineering dimensions".
  kg, m, s (SI)

Relationships between variables imply relationships between dimensions.

Qu. Is there a relationship between kilometres & ~~metres?~~ kilogrammes?

Weight of cable $\propto$ length 

$$W \propto L \Rightarrow W = kL.$$

kilogrammes $\propto$ metres

$$\text{kilogramme} = \frac{\text{kilo}}{\text{metre}} \cdot \text{metre}$$

$$\Rightarrow [k] = \frac{\text{kilo}}{\text{metre}}$$

Packets per year $\propto$ packets / second

~~p/year~~ =

$$\text{packets} = \text{packets}$$

$$\frac{\text{packets}}{\text{year}} = \frac{\text{packets}}{\text{year}} \times \frac{\text{seconds}}{\text{seconds}}$$

$$= \frac{\text{packets}}{\text{second}} \times \frac{\text{seconds}}{\text{year}}$$

Thought – $W \propto L^2$ ever true?

2) Observations + Uncertainties

We observe / measure to understand system behaviour.

- sample, characterize
- identify phenomenon
- formulate hypothesis
- test hypothesis (falsification)

Measurements are

- quantitative (numerical)
- qualitative (classes e.g. red / brown)

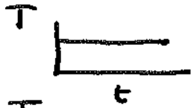
Finite accuracy ⇒ these 2 are not so different.

2.1) Sample / Characterize.

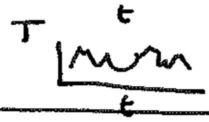
- Take several measurements + see what you get. Repeat until we see a stable picture.

e.g. measure temperature (at different times!)

(i) always get same value



(ii) distribution of values

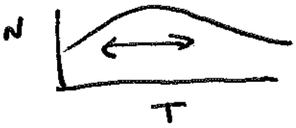


Characterize values by frequency analysis

(i)

same value (low entropy).

(ii)



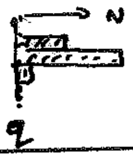
variation. (high entropy)

2.2) Cause + change parameters

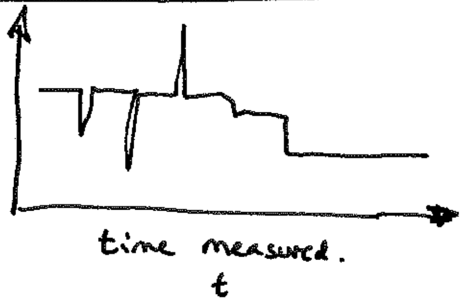
We can only compare values measured under "same" / similar conditions.

e.g. temp on Mt Everest / temp in basement
power used by PC in datacentre / at home

Parameters for measurement e.g. x, t: location, time. Typically time series:



e.g.



This kind of variation is "real", part of the system's true behaviour

PRINCIPLE:

$\boxed{\text{change only one param at a time, to separate causes.}}$

PROBLEM.

$\boxed{\text{our interaction with the system alters it or our perception of it}}$

Need to separate this from 'reality'.

2.3) VARIATION IN MEASUREMENTS

Always interested in change ($q(t)$)

- scatter (variation in q)
- jitter (variation in arrival time of events)

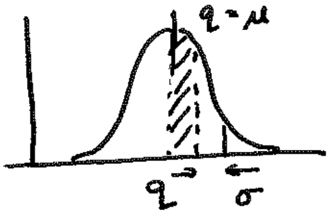
→ also use μ, σ here! more this characteristic value

2.4) MEASUREMENT ERROR

Assume some value q has a fixed value. We cannot know if we measure q! (Sample many times)

(i) Random error.

- External factors cause false variation in value (symmetrical)
- Assume independent causes for each sample.
- Assume variation is Gaussian.



$$N(q) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-(q-\mu)^2/2\sigma^2}$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} q_i \qquad \text{(sample size } n\text{)}$$

$$\sigma = \sqrt{\frac{(q_i-\mu)^2}{n}} \qquad \text{(RMS)}$$

$$\sigma_{n-1} \equiv \sqrt{\frac{(q_i-\mu)^2}{n-1}} \geq \sigma$$

When quoting a "believed value" we write

$$q \pm \Delta q$$

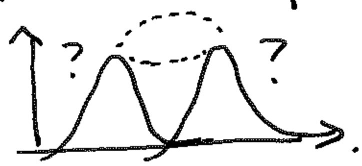
and use:

$$\boxed{q \simeq \mu \pm t\,\sigma_{n-1}}$$

We "believe" μ is the "correct" value but it could be $\pm$. cmp # with characteristic value

t is called student's t-value. See t-test, Welch's test.

t-test: Are these peaks distinct?



(ii) Systematic Error

Measure incorrectly each time

e.g. clock runs fast
rounding error in software

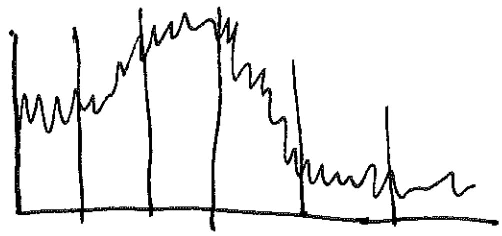
These cannot be eliminated by repetition – must compare theory, expectation, with result.

e.g. "numbers look too big to me"

2.5) CRITIQUE

- Everyone tells you about Gauss, but not everything is "random error"
- Textbooks often assume one true value – but values change for valid reasons, not error.

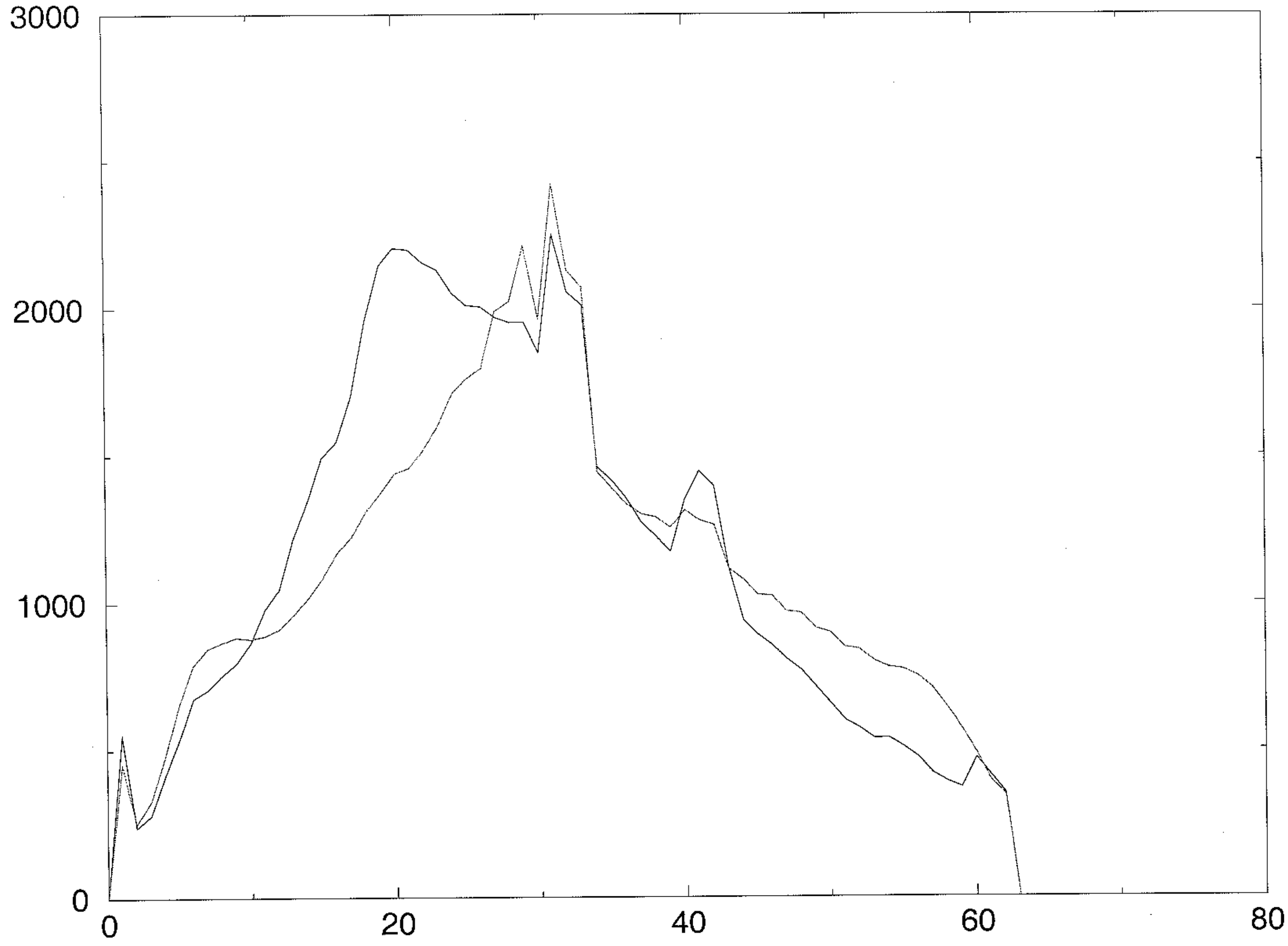
PRINCIPLE $\boxed{\text{Separate slow change from rapid change.}}$



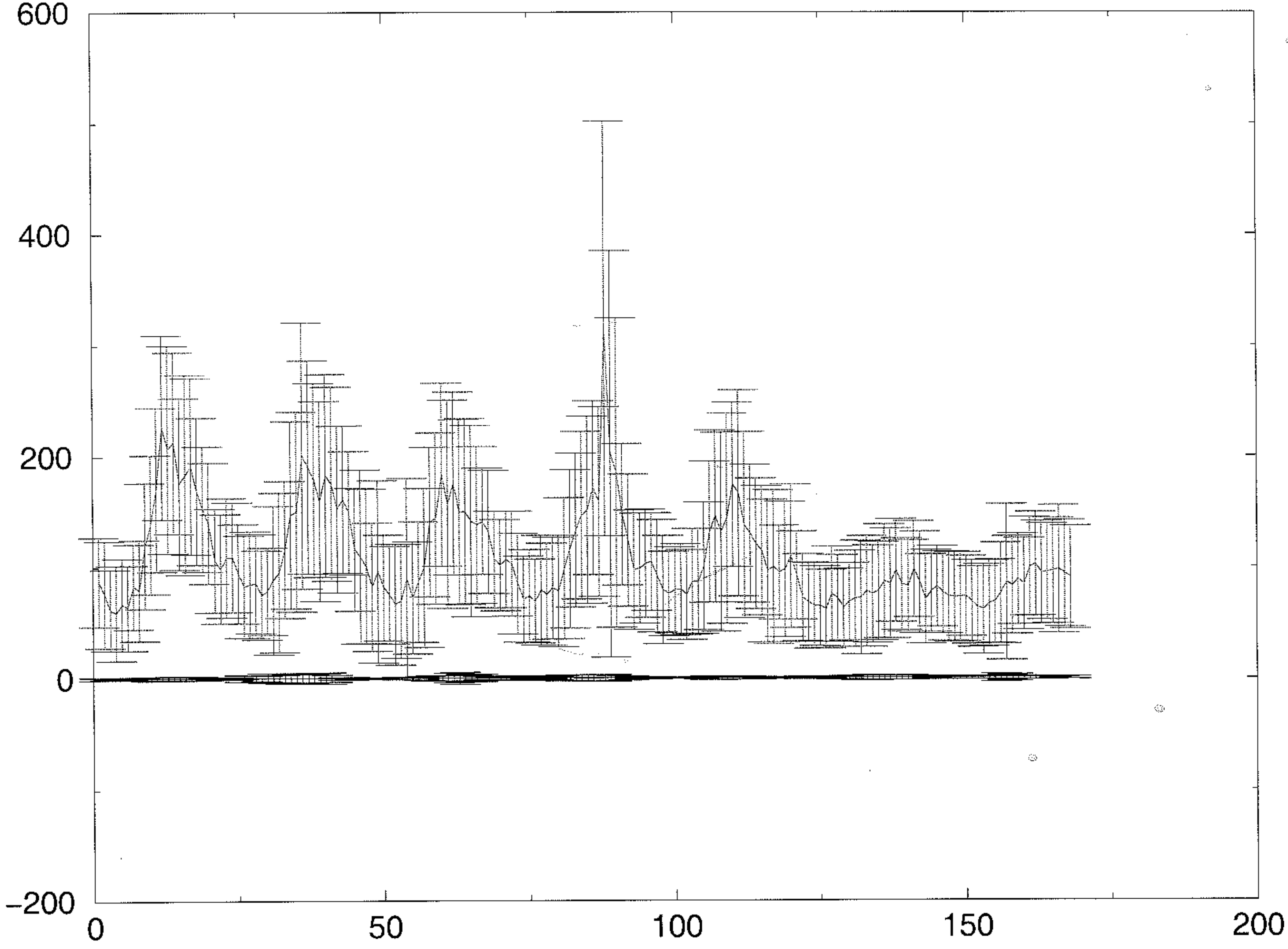
mean = slow value

σ = scale of fast variation.

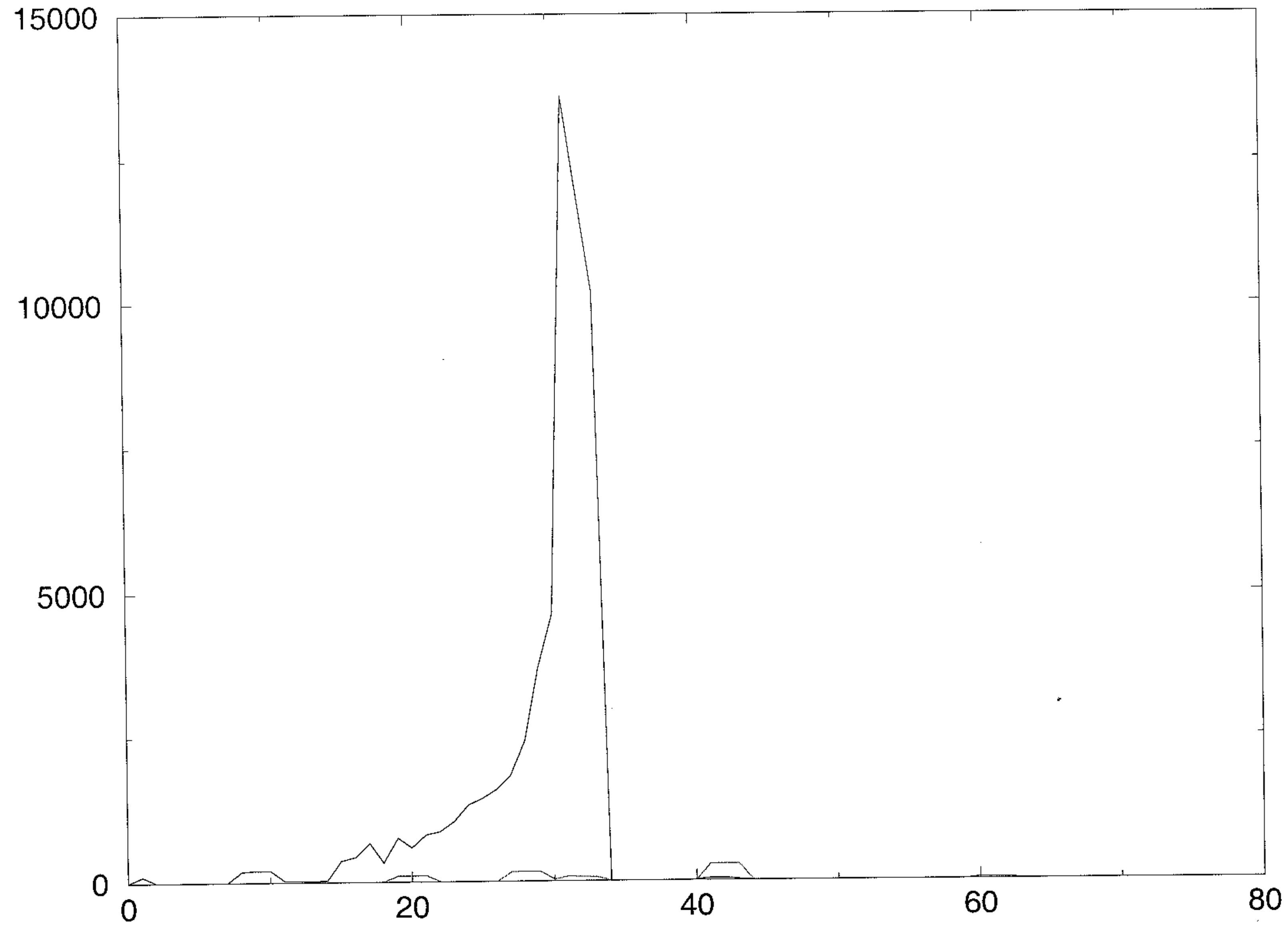
root / other processes dist

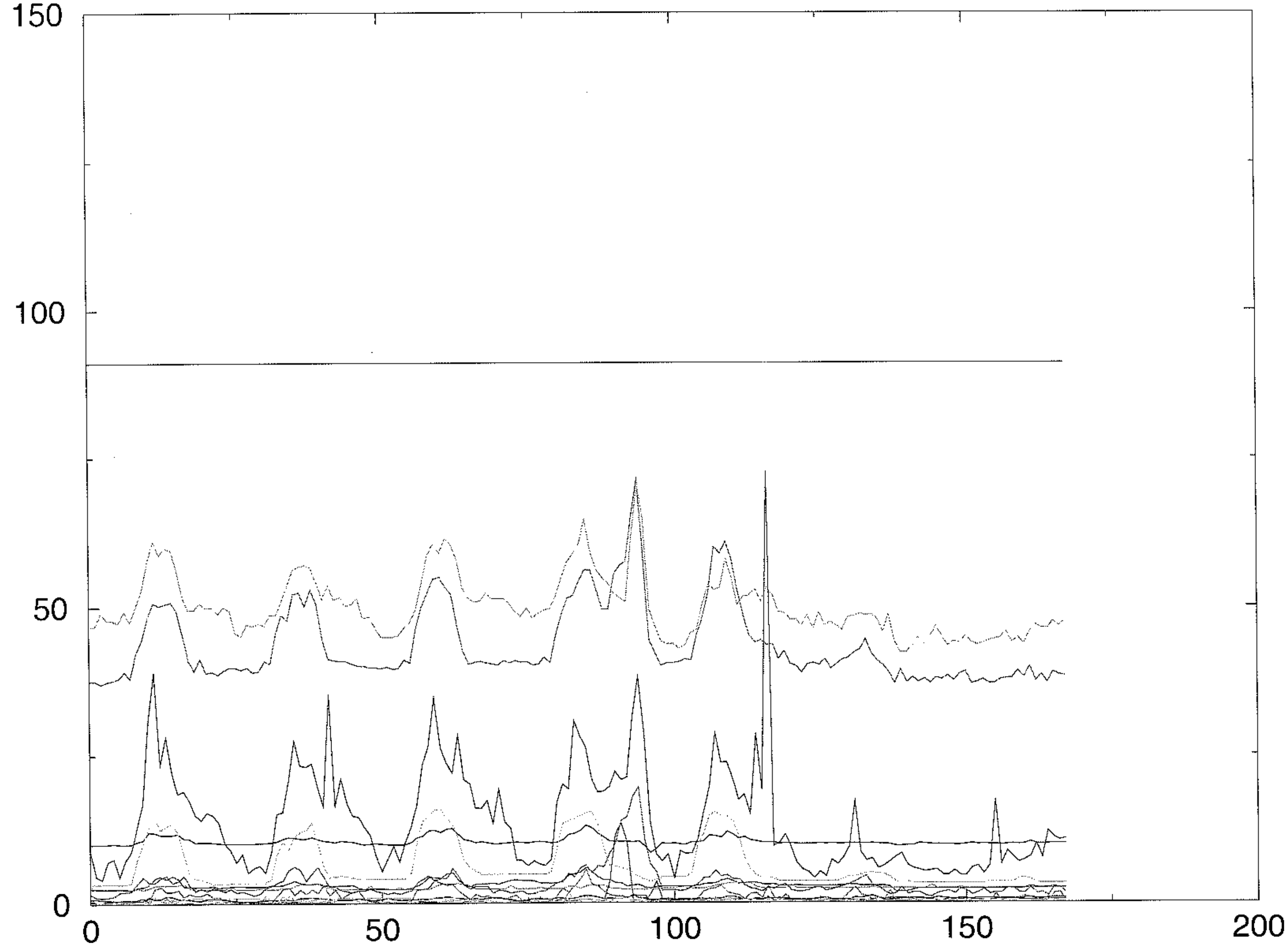


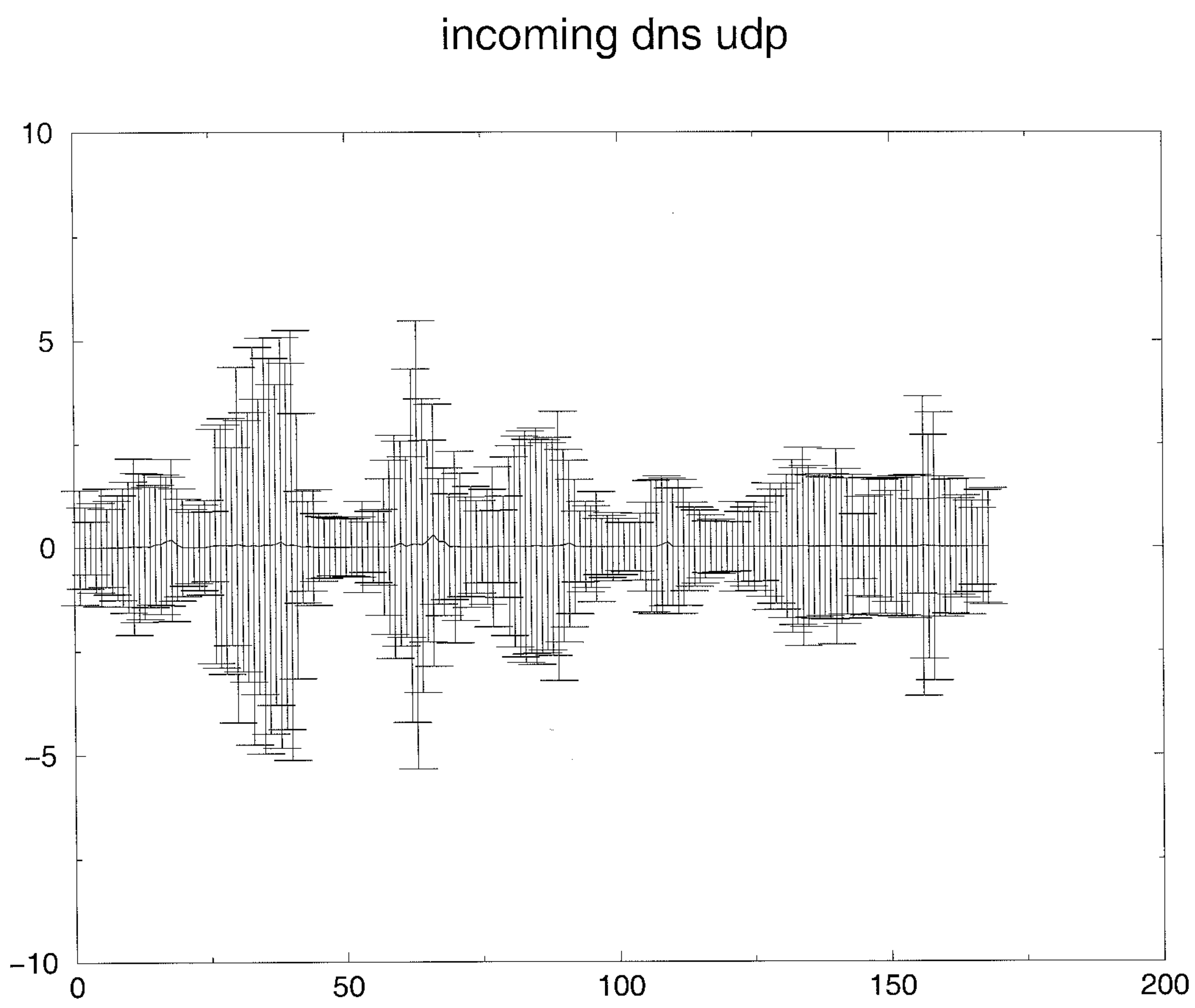
outgoing dns udp (origin iu)



misc net traffic in/out

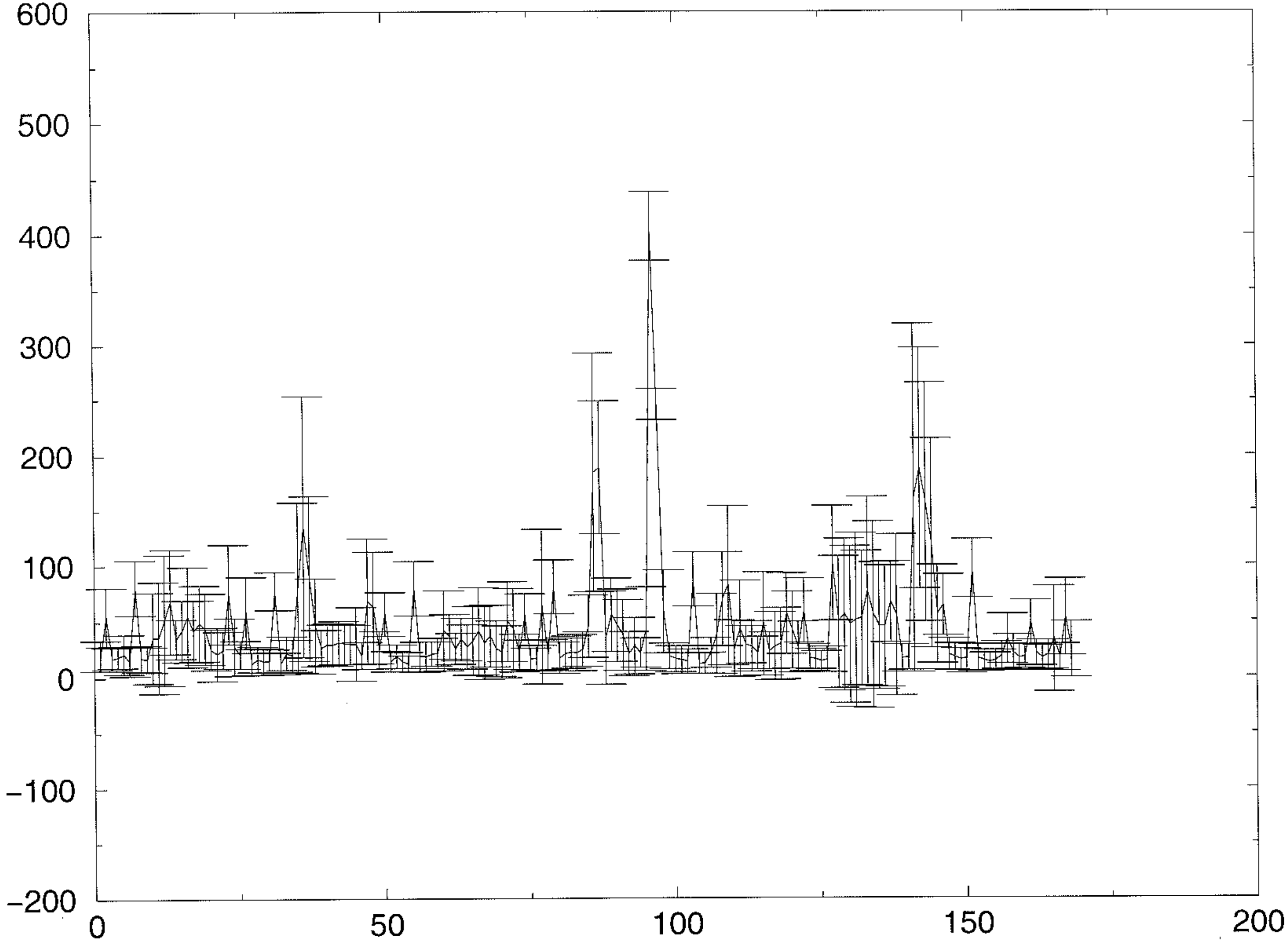



150
100
50
0
0
50
100
150
200

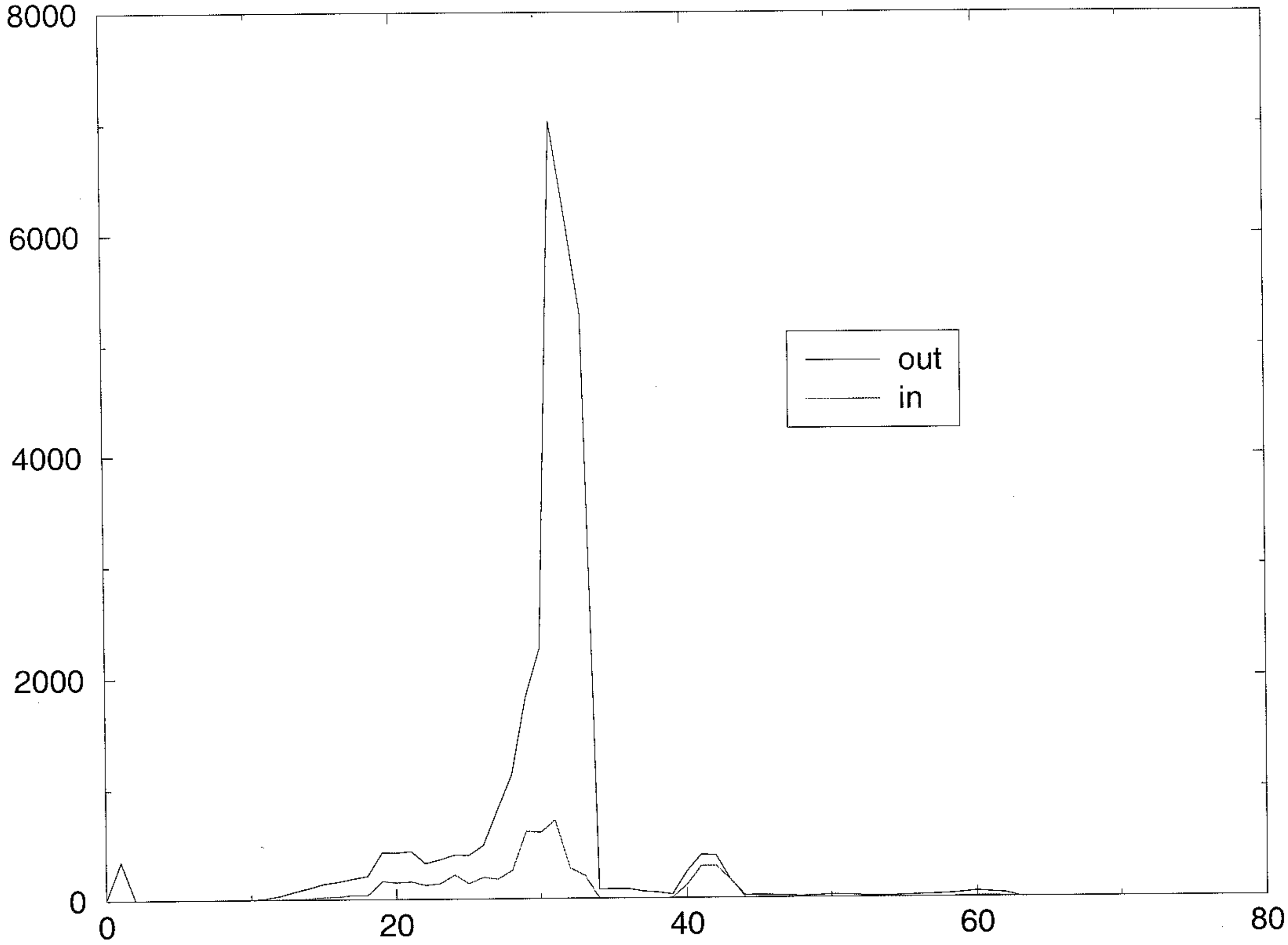


incoming dns udp
10
5
0
-5
-10
0
50
100
150
200

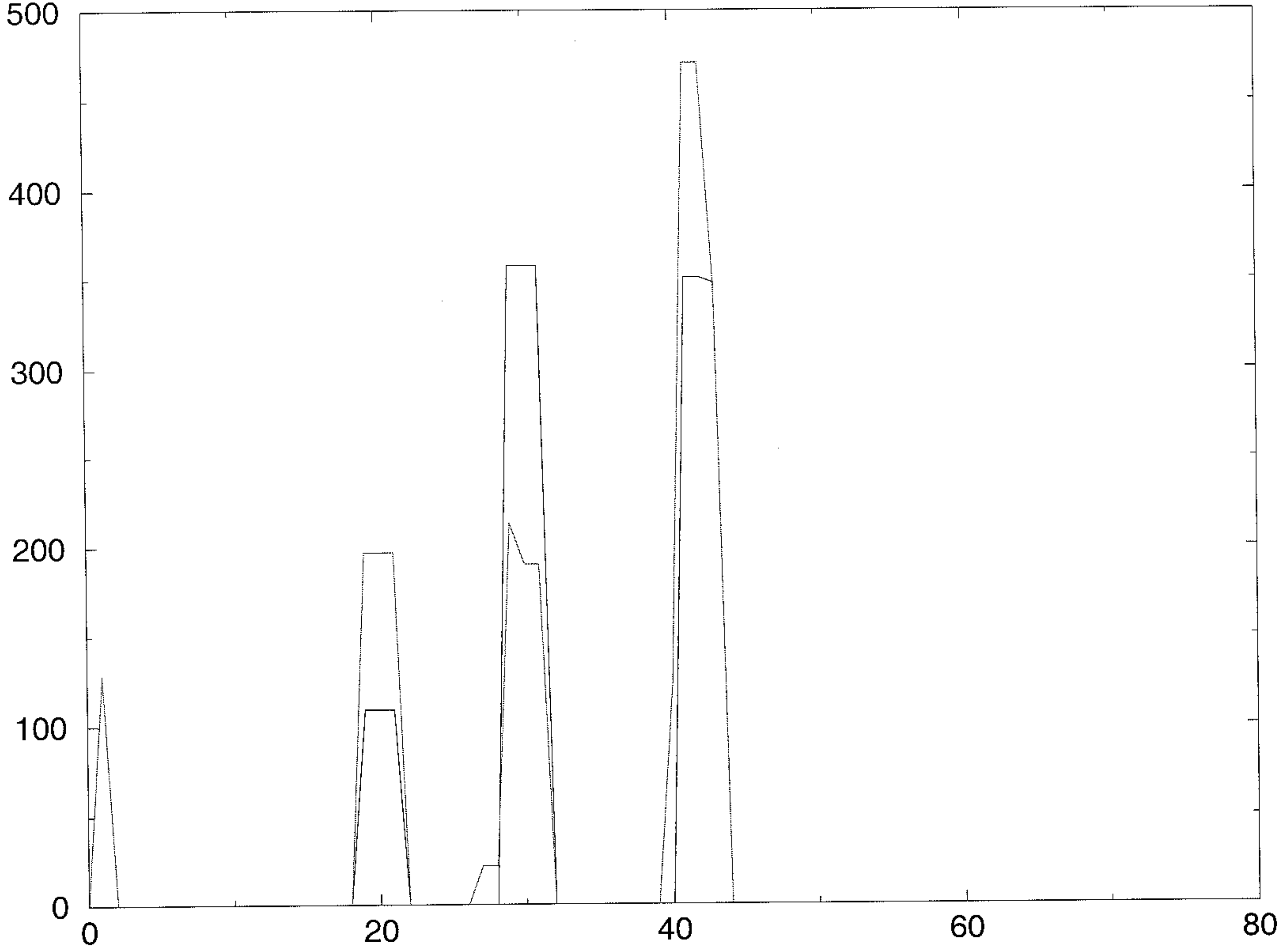
load average week



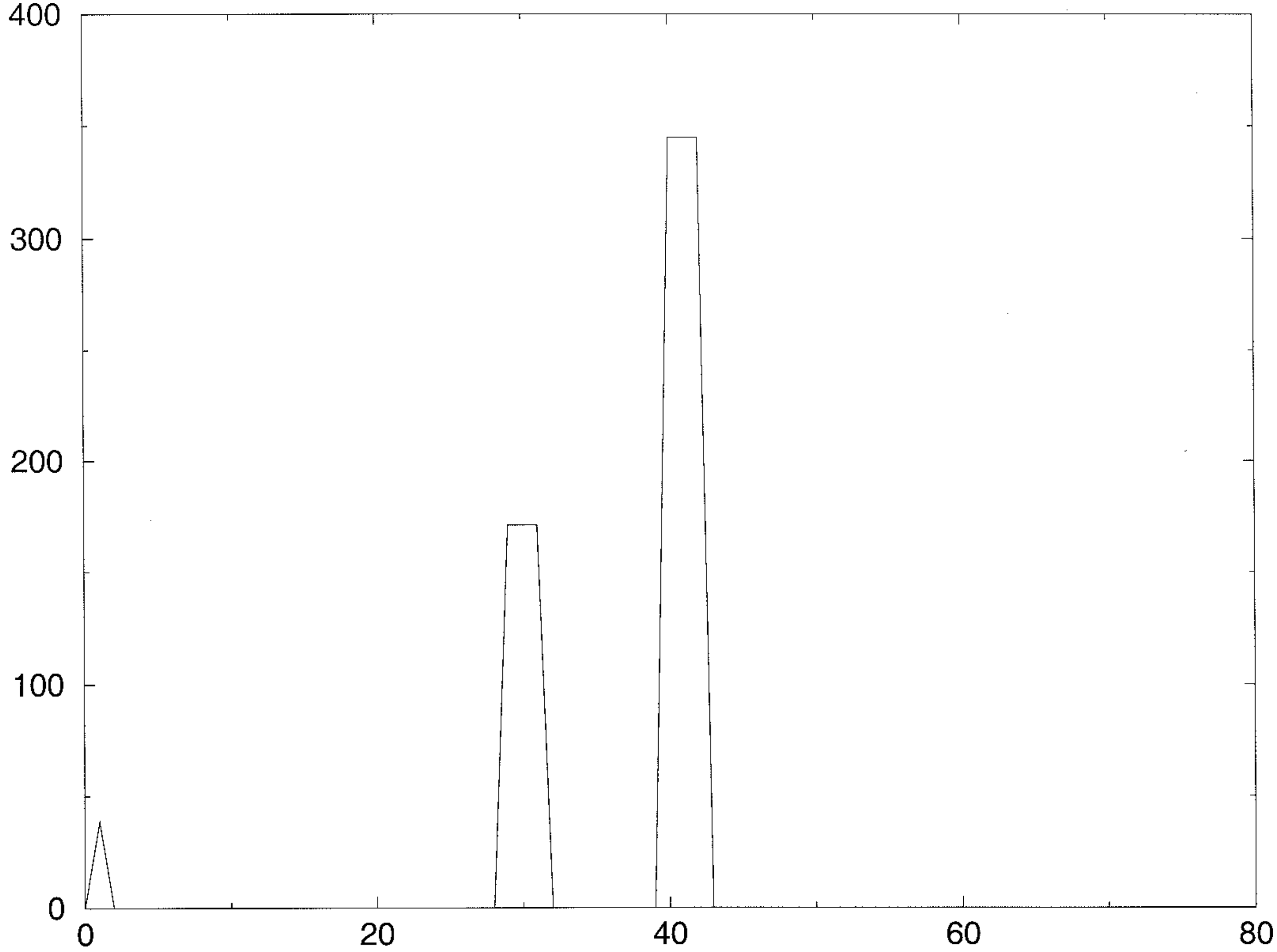
tcp-ack-out / in



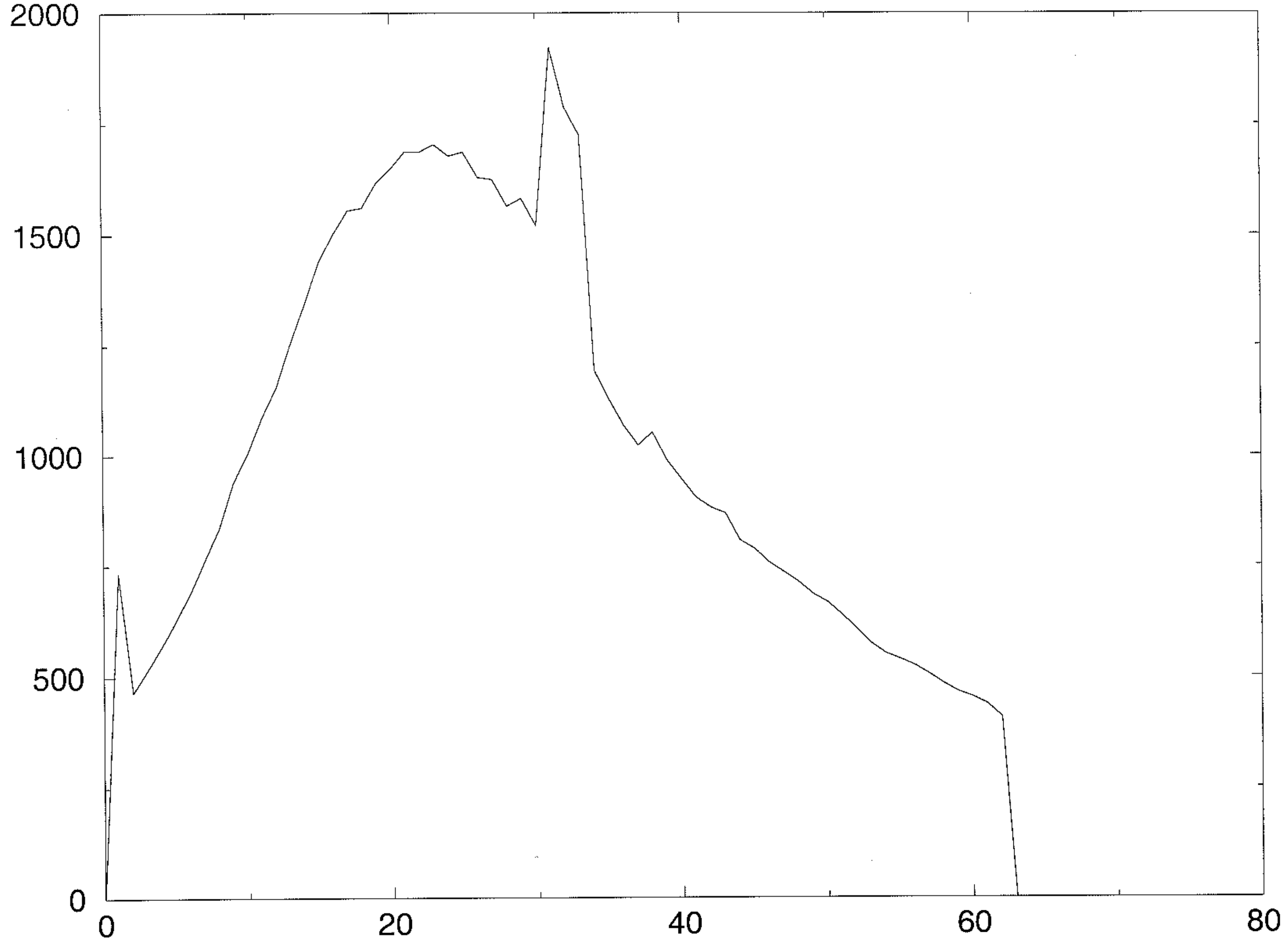
tcp syn in/out dist

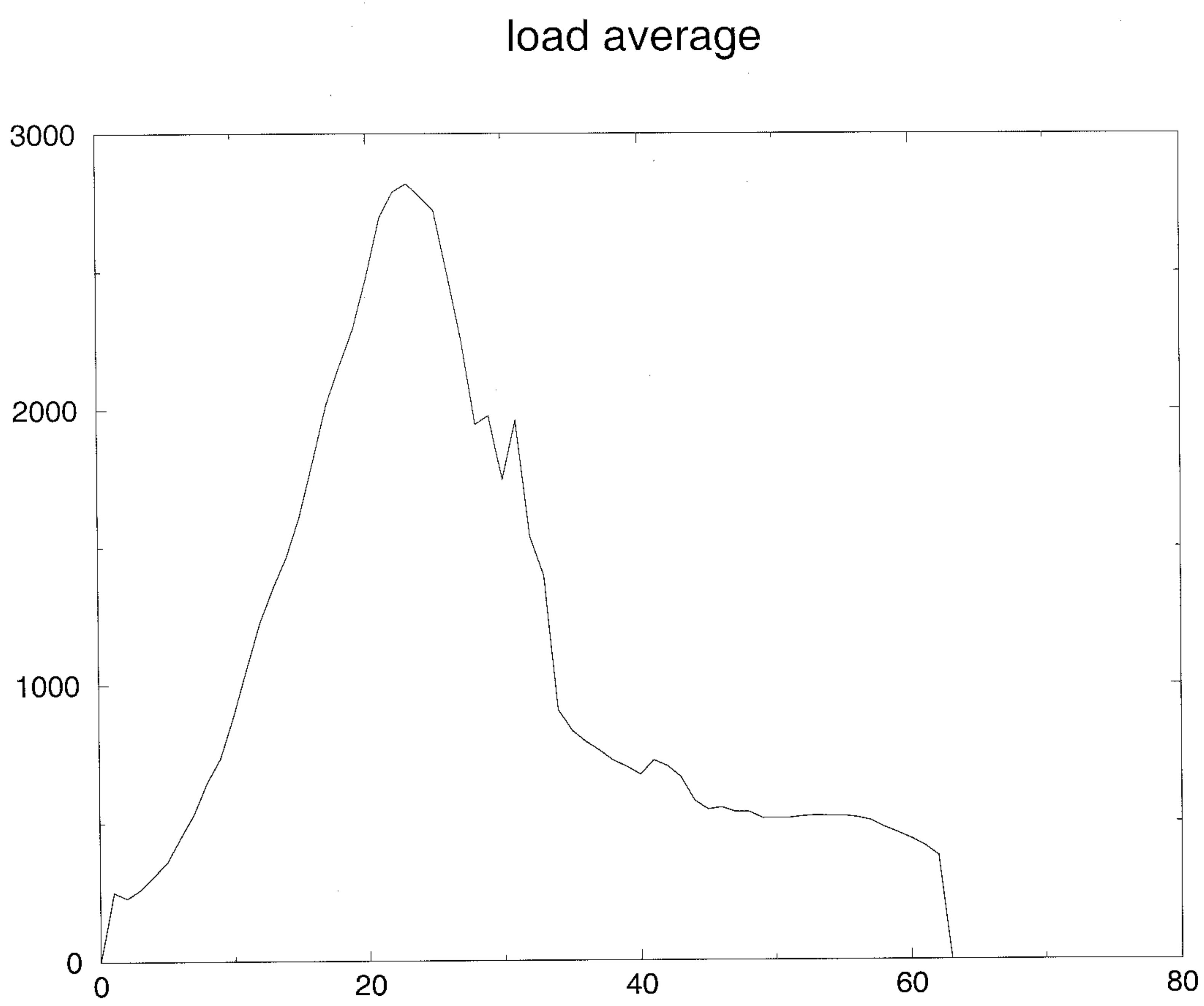


wwws in dist

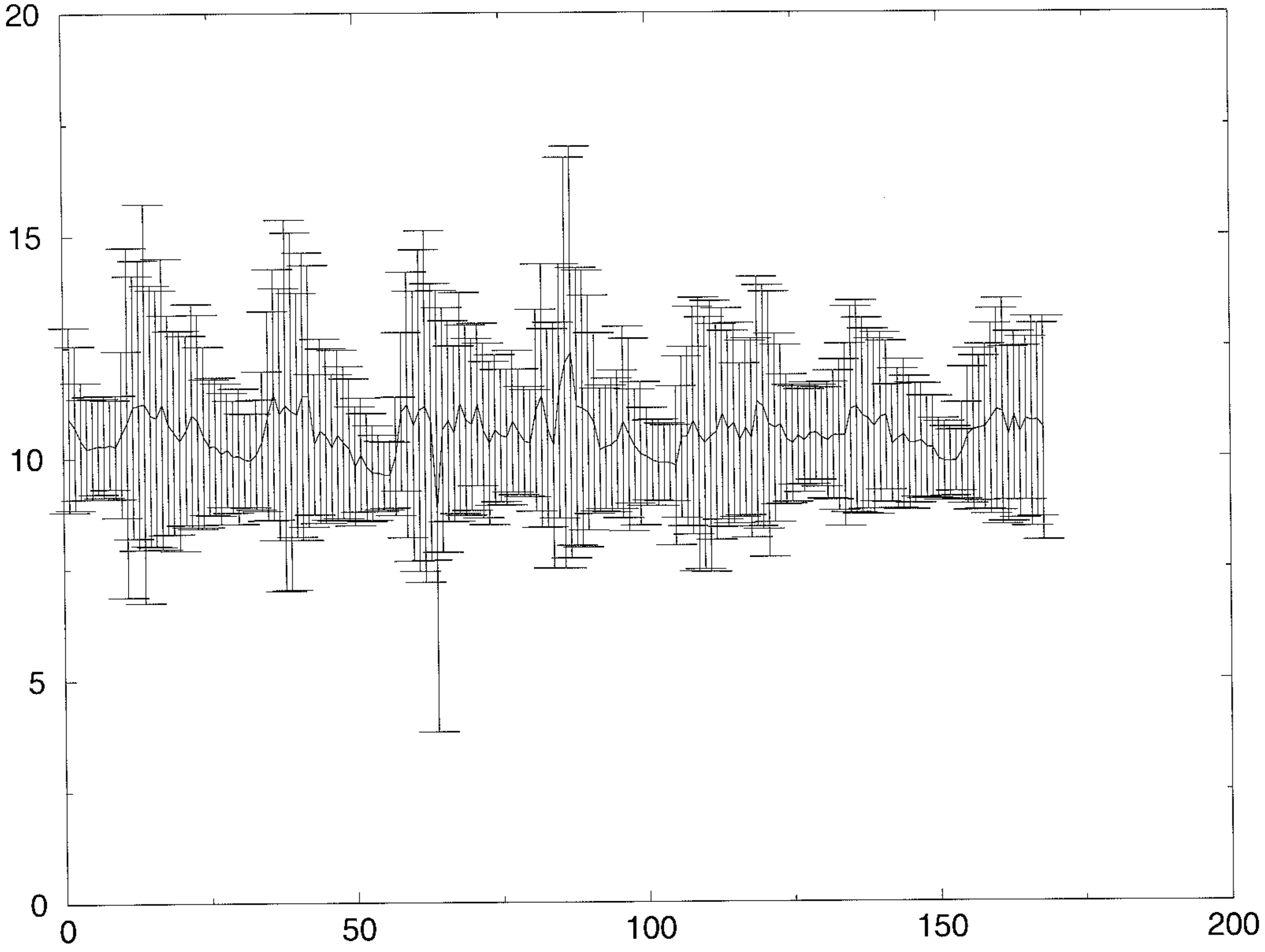


www in dist




load average
3000
2000
1000
0
0
20
40
60
80

users (week)



3) Discrete vs continuous

Last: qualitative, quantitative + uncert. This: change + variation

There are 2 approaches to modelling change or variation

(i) Discrete, countable, distinct number of cases e.g. {windows, linux, mac ...}
describes: jumps

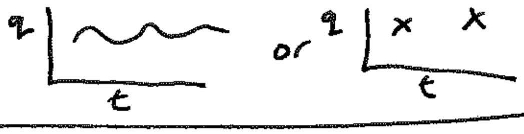
(ii) Continuous – infinitely many variations
e.g. 0.1, 0.11, 0.110136. Smooth trends – describes "morphing"
(always an approximation)

Factoid: is a wave continuous? – Depends how you look at it!
Large scale: wave! Classical E.M.
Small scale: photon: Q.M.

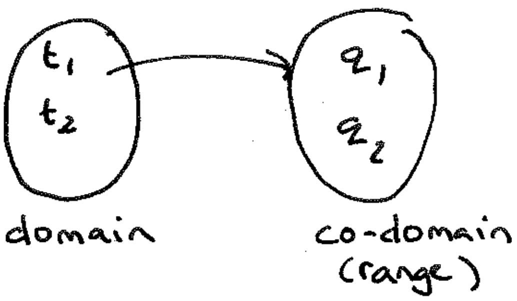
Don't forget we are dealing with models!

Parameterized change

Each measurement, observation or prediction associates a value $q \in \{Q\}$ with a time or place $t, \vec{x}$



Thus each "event" is a mapping



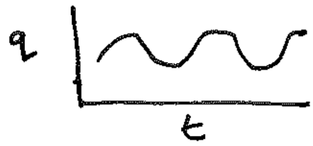
Both domain and range can be independently discrete / continuous.
e.g. $t = 1, 2, 3$ or $0.1, 0.23, 5.6$

$q = 10, 20, 30$ or $14.75, 16.73$

(i) discrete association
e.g. arrays in perl

```
$array[t1] = q1 ;
$array["windows"] = "yippee";
```

ii) continuous association



$q = \sin(\omega t)$

We say that a continuous function $q(t)$ describes such an association.

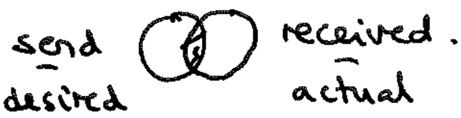
BOTH CASES: mapping between sets.

VENN DIAGRAMS "sets, sets, sets ...."

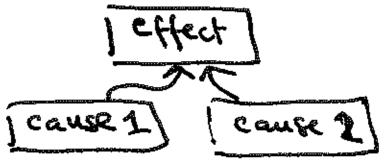
$A \cap B$ – intersection (AND)

$A \cup B$ – union (OR)

Use this in configuration management and communication: overlap!



Probability rules for fault trees (recall computer security, risk mgt).



$$P(1 \text{ AND } 2) = \underbrace{P(1)P(2)}_{\text{overlap}}$$

$$P(1 \text{ OR } 2) = \underbrace{P(1)+P(2)}_{\text{union}} - \underbrace{P(1)P(2)}_{\text{overlap}}$$

$$P(1 \text{ XOR } 2) = \underbrace{P(1)+P(2)}_{\text{union.}} - \underbrace{2P(1)P(2)}_{2\times\text{overlap}}$$

Describing change.

(i) discrete ^transitions^

business, queues, conf mgt

We label the different alternatives as a set of "states".

transition diagram or "finite state machine" or "automaton".

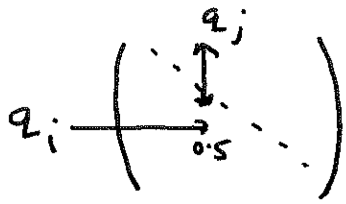
- e.g.
  - q_1 = start,
  - q_2 = busy, (running)
  - q_3 = waiting $\quad q_4$ = done

The state can change when an "event" or "trigger" occurs.

Probability of a transition from $q_1 \to q_2$ is written. (conditional)

$$p(q_2 | q_1) \quad - p_2 \text{ given } q_1$$

$P(q_i | q_j)$ is a <u>matrix</u>



$p(i/j) = 0.5$

If $p(i,j) = 1 \Rightarrow$ "deterministic".

$\sum_i p(i,j) = 1$.

(ii) Continuous transitions

communication, resource economics, averages, performance

- described by differential equations
- e.g. heat equation
  wave equation etc.

We know some continuous patterns already, e.g. sin, cos, log, exp ...

Discrete patterns are made from <u>states</u> or <u>symbols</u>.

Start with an alphabet $\Sigma = \{A, x, \}$

Syntax = list of all valid strings formed from Σ.

LANGUAGES

In all cases we are describing <u>patterns</u> of behaviour in some variable $q(t, x, \ldots)$

Grammar – a set of rules for constructing valid syntax (anal. diff eqns)

The "Chomsky Hierarchy" describes 4- levels of language

1. regular (FSM)
2. context free .. (Push-down)

He showed that there is a mapping between automata and ~~gr~~ grammar.

Patterns of lang. can be parsed (identified) by automata of level n.

e.g. regular expressions

.*

e.g. TCP language.

$\Sigma = \{SYN, ACK, FIN, DATA \ldots\}$

Pattern: [SYN][ACK]* [DATA]* [FIN][ACK]

4) Configuration

chaps (15, 16) 5.5 →

Define config = set of states $\{q\}$.

We know that states are changed by operators (model of change) (matrices)

How do we manage specification of system (promises) and arrange maintenance to keep those promises?

config. hypothesis

$$\boxed{\text{Policy} \rightarrow \{\text{configurations}\} \rightarrow \{\text{behaviour}\}}$$

i.e. we decide behaviour by config / programming.

This is unproven, but it is clearly true to some approx. (NP hard)

Maintenance theorem

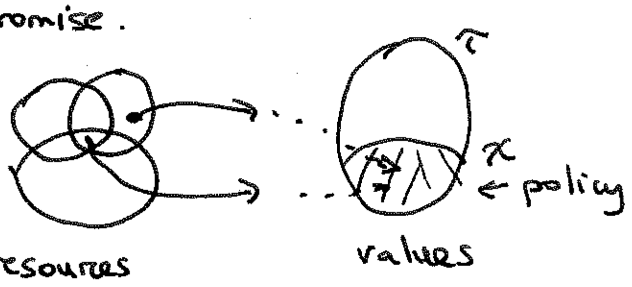
> A limited number of policies lead to configurations that give unique behaviour on average.

i.e. there is a probabilistic aspect.

> Most approaches try to use obligations. unrealistic

Promises.

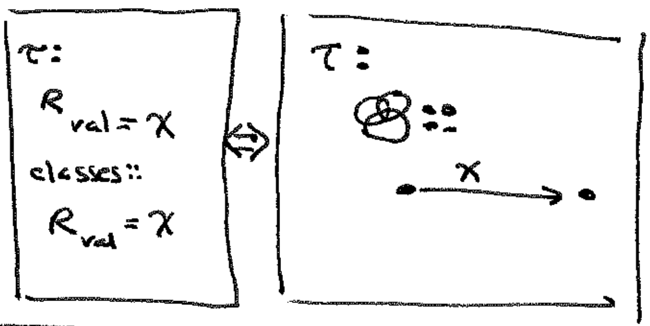
We talk about promises because we do not always need to do anything to keep a promise.



promise that some subset of resources (files, machines, processes..) will restrict their config q to subset $\chi \subset \tau$. Promise:

$$\underset{\text{resource}}{R} \bullet \xrightarrow{\tau, \chi} \bullet \underset{\text{monitor}}{M}$$

Cfengine is an implementation of promises



Operators - generalized matrices.

A promise has to be verified – was it kept? How often, consistently?

- check state
- need change?
- act (relative or absolute goal).

For each promise

$$R \xrightarrow{q = \chi \subset \tau} M$$

we want an operator that "keeps" the promise when applied. (a service)

$$\hat{O}_{\tau}(q = \chi)$$

- How long will it keep the promise?
- How often should it be applied? (service level agreement).

Two strategies for operators

- Known path. Know q_{start} and q_{end} and specify

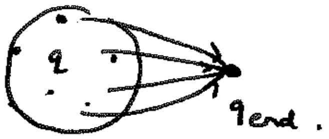
$$q_s \rightarrow q_1 \rightarrow q_2 \cdots \rightarrow q_{end}.$$

(congruence)

This is fragile to unknown change in q_s + order dependent

- Known destination

Don't care about q_{start}. Find a way to get to q_{end} and stay there. (convergence)



operators:

$$\left.\begin{aligned} \hat{O}\, q_s &\rightarrow q_{end} \\ \hat{O}\, q_{end} &\rightarrow q_{end} \end{aligned}\right\} \quad q_e = \text{"fixed point"}$$

This is robust to random changes in q_s. Just keep applying same operation.

Idempotent or convergent?

Couch et al. have suggest that

$$\hat{O}^2 = \hat{O} \qquad \text{is enough.}$$

but this is not tied to a specific q_{end}. Convergence knows about a specific end state.

e.g. $\hat{O}_i$ = create ("myfile")

$\hat{O}_i^2$ = create ("myfile")

cd different directory?

convergent:

$\hat{O}_c$ = create ("~/mydir/myfile").

$\hat{O}_c$ is also idempotent. i.e. convergence is idempotenence at a fixed end-point.

Ordering + orthogonality

Linear independence, orthogonality like vectors.

- A change in one property does not affect anothe property.

e.g. $\hat{O}_1 \hat{O}_2 = \hat{O}_2 \hat{O}_1$ (path independence)

$\hat{O}_1$ = chmod (attr 1)

$\hat{O}_2$ = chown (attr 2)

e.g. cfengine 3.

```
type:
   context-classes ::
      "object" -> "monitor",
          attr1 => x1 ,
          attr2 => x2 ,
            :
          attrn => xn ;
```

5) Systems (chap 4)

What makes a system?

> An organized effort to fulfill a function/task (humans + computer)

organized ⟷ structure?
Function ⟷ tools/value?

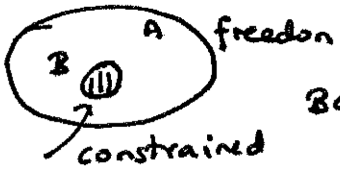
Or do we really mean 'to keep a promise'?

We know patterns are key to describing structure + changes of state.

Organization

The ability to create structure
- freedoms (degrees) (possibility to change)
- constraints (limitations ")

q:



$B \subset A$

Behaviour of a system is observed outcome of these competing issues.

Rules ⟷ policies are constraints (self-imposed)

e.g. config mgt:
freedom: to change files, processes
constraints: allowed perms, contents, processes.

e.g. net.
freedom: services, www etc.
constraints: access controls

Freedom:	Give / Recieve	alphabet
Constraint:	restrict	syntax.

Static or dynamic?

Organization can be static (library or archive, data structure)

or dynamic (operating system, input/output, process start/stop)

Flow of "control"

Who drives change in a system, decides constraints?

Obligation

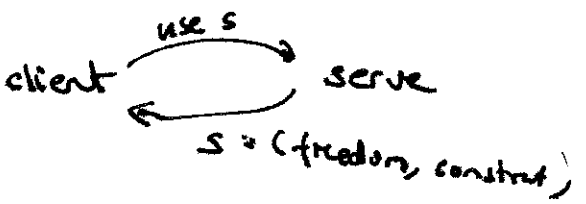
$\text{master} \xrightarrow{\text{oblig}} \text{slave}$

Voluntary cooperation

$\text{client} \xleftarrow{\text{promise}} \text{server}$

Free market thinking.

Client cannot force server to provide
Server cannot " client to use

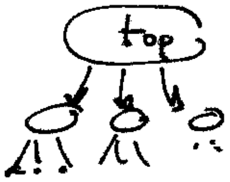


s = (freedom, constraint)

In a service model, entities communicate behaviour by making promises.

Predictability / information

Hiearchy + obligation is the OO model.



"need"
Driving info at top, skills at bottom.
Centralized bottleneck.

Service oriented (P2P)

"need" distibuted.
decentralized.

<u>Centralization</u> makes consistency easy – info in one place but concentrates resources.

Fragile to communication failure
+ conflicts of outside constraints

<u>Decentralized</u> makes consistency harder, but conflicts easy.

e.g. <u>obligation</u> (constraints distributed)

• paint house red → painter.
• paint house blue →

<u>voluntary / promise</u>

→ paint house red.
→ paint house blue ?

(constraints localized – easy)

Tool / function ⇒ value

Technology = creative use of knowledge to add perceived value to humans.

We can use promises together with value estimates to understand why entities might cooperate.

V_{client} (server $\xrightarrow{P}$ client)

V_{server} (~~state~~ $\xrightarrow{P}$ client)

V_{client} (client $\xrightarrow{\text{use } P}$ server)

V_{server} (client $\xrightarrow{\text{use } P}$ server)

This shows the basic relationship between systematic behaviour and commerce.

Question: what is the currency of these values?

$\boxed{\text{TO DO}}$

All exercises in ch. 7.

7) GRAPHS + NETWORKS Chap 6, 11

Continue discussion of organization

Graphs easily represent relationships in systems of discrete objects.

e.g. relational db (ER)
hierarchies (trees)
webs

Relations can be
directed $\rightarrow$
undirected $\leftrightarrow$ OR —

Diagrams are systems

freedoms; space, colour, shape, direction
constraints: relationships, finite resources

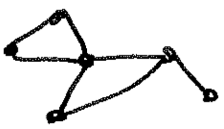
Relationships can be abstract

$\rightarrow$ depends on, accepts from
promises, uses
is greater than (ordering)

$\leftrightarrow$ is next to (neighbour) adjacent.
is correlated with
is connected to

Graphs

A graph is formally a pair of sets. (E, N) of edges and nodes.



The degree (k) of a node is the number of edges (links) it has, e.g.

 $k=4$

 $k_{in} = 2$
$k_{out} = 1$

Graphs are easily represented as adjacency matrices

0 = no edge
1 = edge (neighbour)
p = prob(edge) – ad hoc net

e.g.

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

A is symmetrical ($\searrow$) if undirected.

A is counts neighbours.

Connectivity

Measure the extent to which a graph is fully connected (complete)

 – a "mesh" "complete"

$N(N-1)/2$ undirected links.

Let $\vec{h}$ be a vector of 1's which represents "presence" of nodes.

$$\vec{h} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Define the fraction

$$\chi = \frac{1}{N(N-1)} h^T A h = \begin{cases} 1 & \text{if complete} \\ <1 \end{cases}$$

Use this later to discuss reliability.

Importance ranking

- Like maxima + minima for graphs
- "centrality" in a graph can be used to find importance to connectivity by "voting".

An important node is one that is connected to many important nodes!

nodes n_i

Importance(i) $\to I_i$.

$$I_i \propto \sum_{j \in \text{neighbours}} I_j$$

$$I_i \propto \sum_j A_{ij} I_j.$$

$$\boxed{A\vec{I} = \lambda \vec{I}}$$ for some const. λ.

This is the eigenvalue equation. $\vec{I}$, the importance vector ranks the nodes if it is an eigenvector of A.

"Most connected" node is that which

- best spreads viruses
- likely overloaded (bottleneck)

(Examples)

<u>Percolation</u>

What if we only know probable number of neighbours? (Not number of nodes or exact structure).

Prob. distribution of degree $P(k)$

$$\sum_k p_k = 1$$

$$\langle k \rangle = \sum_k k p_k$$

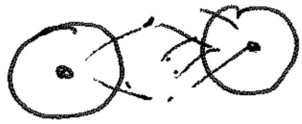
Estimate N = no. of nodes.

$$\langle k \rangle^2 + \sum_k k(k-2)\, p(k) > \log N$$

see (11·6).

<u>Self-organization</u>

Neighbourhood relations (dynamic) cluster around local maximum



(8) A model of human-computer systems

ch. 14, 15 (review)

Look at how analytical methods address 'key areas' of human-computer systems. (current art)

Planning (human)
↓
Deployment ↑
↓
Maintenance (human / auto)
↓
Change ↓

Grammars / operations

- pattern description for config mgt.
- patterns of operations

Convergence

- stability
- reliability, predictability.

POLICY

promises →

↑ operations / change

compliance ↙

monitor

ITIL / etoM / COBIT – "best practices"

- Control / management
- Asset / audit guidelines

control objectives 4 information tech.

Ask humans to work more predictably and document work for error recovery.

Blame ⟷ accountability

"don't blame it on the good times...."

Change ⟷ "repurposing" (virtualization) or maintenance.

Tools we have examined:

Scales / measurements

- capacity planning (calc. availab.)
- service level management (provisioning)
- optimization.

Uncertainty / probability

- error margins
- risk analysis

Discrete models / graphs

- context (relationships, dependencies)
- organization
- bottlenecks
- service promises, consistency
- asset management

Human processes

Humans are often unpredictable. A small amount of discipline can greatly improve predictability.

e.g. cfengine
ITIL

e.g. Menus versus language
More constraints, less freedom, easier to predict.

Information models

Bureaucracy is an important tool going back to Ottoman Empire
– it is how humans remember + process in a pattern of operations

$\hat{O}_{\text{Boss}} \; \hat{O}_2 \; \hat{O}_3 \; \vec{?}_{\text{document}}$

- Factory processing of memory.
- The forms + templates play a role.

This view still dominates (network).
management.
- MIB - (snmp) mgt info. base.
- CIM - common information model
- SID - shared information model/db.

Den-NG
↳ data model (OO) deluxe for policy-based-management.

Change "repurposing" = NORMALIZATION identification + separation of concerns

Autonomics

"autos" - self
"nomos" - the law.

"self-governing" system.

Try to take humans out of the "do loop".
- automatic repair (cfengine)
- life signs
  - pulse
  - brain, independent cells.

- react more quickly if brain decoupled!
- context awareness, adapt to environment.
- ~~keep~~ dedicated service, free rest of system to do its work. in good health.

Summary
- separation of concerns
- virtualization, parking spaces
  - easy re-use.

1. group similar things separate different things
2. Extract sub-patterns e.g. name-services. directory.
3. Avoid common dependencies (strict tree).

- predictable changes + state maintenance.

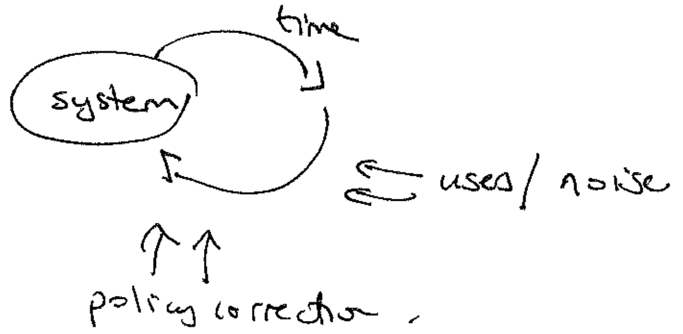
8) Model of Human-Computer System

Put together some ideas.

- satisfy objectives / constraints.
- maximize return on investment (profit)
- Cannot predict demand / load
- Noisy environment means state of systems is being corrupted. (unpredictability)

Maintenance model / config mgt

- think in terms of operations
- change / repair / implementation
- sequences / types determined by <u>policy</u> <u>constraints</u>



Can think of all processes as communication

- of current state (noise → error)
- of policy state

(noisy channel).

e.g. think of maintenance as the transmission of symbols associated with

convergent operators.

$$\hat{A}\hat{B}\hat{C}\hat{D} \rightarrow \hat{A}\hat{X}\hat{C}\hat{D}$$

↑ correct.

Graphs / Networks

- Architecture, interactions
- operations become <u>services</u>

constraints → SCA.

Digressions

Next week — more on communication.

+ networks.

①

8) A model of human-computer system

ch ~~##~~ 115

Let us put together some of the ideas thus far to make a picture of a system. We could wait until the end, but this is a half-way check

- We think of a human-computer system (a business f-ctr) which has processing objectives.
- Looking for ways to maximize "return on investment" or reduce "cost of ownership" (for people + equipment).
- We call this "rational" management.

Align goals (business) with

System has __freedoms__ (needs)

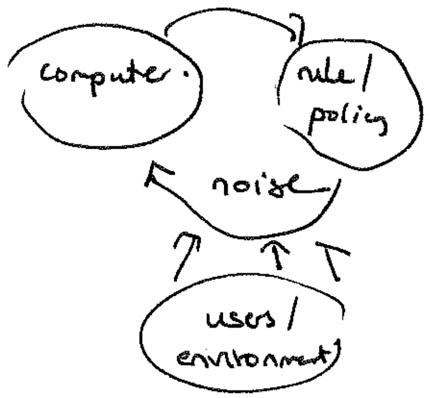
cpu, memory, disk, human time, flows, information etc..

(__things we can change__)

__Constraints__ (things we must not change).
(debts)

hardware limits,
business goals. Costs < profits.
service-level-agreements (SLA),
Promise graphs / SLA. Dependencies (13.4)

The system evolves in a "noisy loop"



__State__

- configuration
- bank balance

Changed by __operations__ $\hat{O}$.

__Policy__

Average state should change slowly - (Requires maintenance).

__Signal + noise__ (Discrete or continuous)

Think of our state as a signal that we are sending into the future.

Unpredictability ⇒ errors / faults changes

⇒ state is unpredictable (noise).

⇒ Need to do correction / maintenance (15.5)

We can identify states with the transmission of ~~operato~~ convergent operators, expressing policy

$\hat{A}\ \hat{B}\ \hat{C}\ \hat{D}$

Noise / errors mean that

$$\hat{A}\hat{B}\hat{C}\hat{D} \longrightarrow \hat{A}\hat{X}\hat{C}\hat{D}$$

Need to correct this.

Since $\hat{B}$ is convergent, just send $\hat{B}$ again

e.g. $\hat{B}$ = chmod 644 /etc/passwd

$\hat{X}$ = chmod 664 /etc/passwd

How do we do this for continuous / performance anomaly?

__Graphs__

Top down / bottom up design (13.5)

Service model (continuous)

Admin can be thought of as a service, with certain QoS goals.

- We need to do maintenance at a certain rate.
- We need to minimize costs and maximize efficiency of IT services to make return on investment.
- Random arrivals (a queue of jobs to perform

(Need to study queues more).

Estimates, dimensioning based on observation.

Need expectation values over time based on measurement

e.g. income from a web service.

$$\langle I \rangle \propto \underset{\text{customer}}{\langle N \rangle} \underset{\text{rate}}{\langle \lambda \rangle} \underset{\text{price}}{\langle P \rangle} \Delta t$$

$$[1] \left[\frac{\text{Trans}}{\text{sec}}\right] [\text{kr}] \ \text{sec}.$$

= transaction kroner

$$\langle \text{Costs} \rangle = \langle \text{purchase} \rangle + \langle \text{monthly} \rangle \Delta t$$

Want $\langle \text{Income} \rangle \geq \langle \text{Costs} \rangle$

Efficiencies (Continuous error detection)

Design decisions affect performance. (Rates - we understand through continuum models).

Performance tuning can only be done if we understand causality through a quantitative model.

Next

Processing of queues + errors.

Exercises this week

Estimating capacities, modelling the value of something for our policy.

Linear algebra - helps you to see the properties without guessing

1. Suppose we have $\hat{O}^2 = \hat{O}$

Then $\hat{O}\psi = \psi'$ (on some state ψ, ψ' unknown).

but we know $\hat{O}^2\psi = \hat{O}\psi = 0$

Suppose:

$$\hat{C}\psi = \psi_0 \qquad (\psi_0 \text{ known}).$$
$$\hat{C}\psi_0 = \psi_0$$

act with $\hat{C}$ on first

$$\hat{C}^2\psi = \hat{C}\psi_0 = \psi_0 \quad (= \hat{C}\psi!)$$

~~and again~~

$$\cancel{\hat{C}^3\psi = \hat{C}^2\psi_0 = \hat{C}\psi_0 = \psi_0}.$$

$\Rightarrow \hat{C}^2 = C$ but more $\to \psi_0$ only

$\alpha \oplus \beta = \beta \oplus \alpha$?

---

1 hydro.
2. Telenor cable.
3.
:

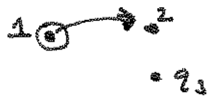
$$\left.\begin{array}{l} PUT = PUT^2 \\ GET = GET^3 \end{array}\right\} \text{assuming constant. read/write?}$$

6) Configuration Mgt

chap 15, 16
(see popular too)

We know that states are changed by operators - This is our model for change.

Let $\vec{q}$ be a state vector and $\hat{O}$ be an operator. Example, representation



$$\hat{O}\, q \to q'$$

$$q_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \hat{O}_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\hat{O}_{12} q_1 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \vec{q}_2{}'$$

We want to manage the state of system resources (disk, memory, devices).

Policy is represented as a choice of which operators to apply in a given state.

i.e. observe state, check if agrees with policy, if not apply operations. (Maintenance)

Main hypothesis

> There exists a policy such that there is a set of one or more equivalent configurations which lead to desired behaviour.

"configuration" means "state" of system resources, expressed in some Language L. (i.e. coding).

This hypothesis is unproven.

So far we have only "the maintenance theorem".

> A limited number of policies can be identified with configurations that lead to unique behaviour on average.

Policy $\rightarrow$ Configuration $\rightarrow$ Behaviour.

We know this problem is NP hard.

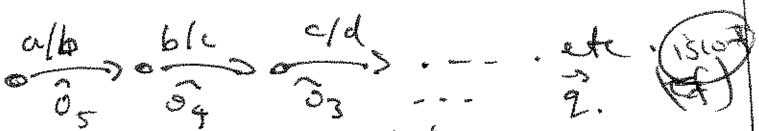
Two strategies

1) Start from a known state, apply a set of changes; ~~until~~ assume the desired state is reached.
(preconditions - each operator promises π_i iff precondition true

2) Start in any state, apply changes until we reach desired state
(post condition - operators promise π_i iff not post condition.

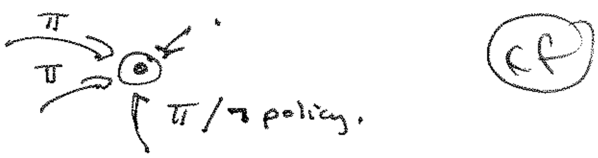
Called i) congruence (need start and final state) (isconf).
ii) convergence (Cfengine)
(must know about final state)

i) is a chain of promise a if b (a/b)



$$\xrightarrow[\hat{O}_5]{a/b} \circ \xrightarrow[\hat{O}_4]{b/c} \circ \xrightarrow[\hat{O}_3]{c/d} \cdots \quad \text{etc} \quad \vec{q}.$$

ii) is an absorbing state



Problem (worse in (i))

If order of operators matters, small errors can break the chain (fragile).

e.g. $\hat{O}_{23} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$

$$\hat{O}_{23} \hat{O}_{12} q_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \vec{q}_3$$

$$\hat{O}_{12} \hat{O}_{23} q_1 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$= \vec{0}$$
$$\neq \vec{q}_3$$

We say O_1, O_2 do not commute

$$\hat{O}_1 \hat{O}_2 - \hat{O}_2 \hat{O}_1 \neq 0$$

6.2

Order dependence is bad because we can get stuck. How can we avoid this?

If operators ~~never~~ are orthogonal they change non-overlapping sets of states within configuration.

e.g. $\hat{O}_1$ = "chmod 644"

q = /etc/passwd.

$\hat{O}_2$ = "chown root"

$$\hat{O}_1 \hat{O}_2 q = \hat{O}_2 \hat{O}_1 q.$$

(only post conditions here)

Fixed point convergence (cfengine).

Avoid preconditions!

Look for a set of orthogonal $\{\hat{O}\}$ operators that satisfy,

$$\hat{O}\vec{q}_{broken} = \vec{q}_{fixed}$$

$$\hat{O}\vec{q}_{fixed} = \vec{q}_{fixed} \qquad \text{(convergence)}$$

This is hard to achieve because we don't control all aspects of an operating system. Sometimes the best we can do is

$$O^2 = O \qquad \text{(idempotence)}$$

Difference.

$$O^2 q = O q = p$$

$$O^2 q = O q = q_{+1} \quad \text{(unpredictable) has preconditions)}$$

Controversy: is ordering necessary?

Graphs

Note that each transition or promise of transition forms a graph.

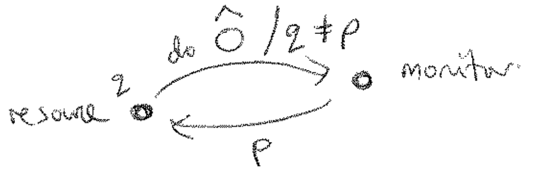
We shall show that a graph is an operator for distributed policy.

Cfengine

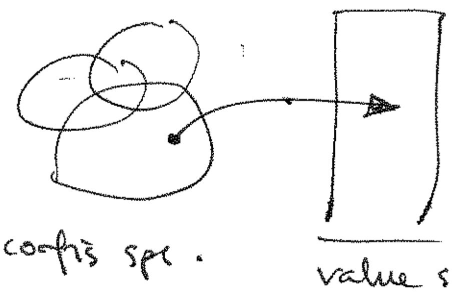
promises: classes:

"object"

attribute → value



promises.



Each promise is associated with an operator that represents a way of keeping the promise.

10) Service Queues chap 12.

We saw that once the number of waiting requests passes a certain limit, we never recover.

- Each communication forms a queue:

if use-rate < request rate it seems logical that the number of waiting requests will grow.

$\boxed{\text{In fact much worse than this!}}$

Arrivals do not come in nice uniform pattern.
- "stochastic arrival process"
- average rate
- also processing rate is not constant
- time to complete is stochastic!

Queues

Classify different processes in Kendall notation.

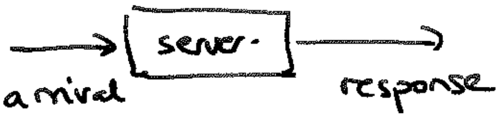
$$A/S/k.$$

A = arrival distribution

S = service processing dist.

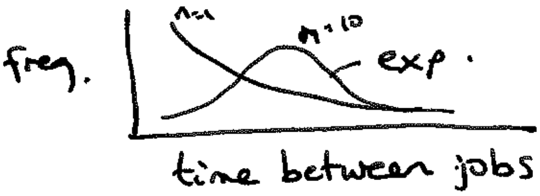
k = no. of servers.

Look for (predict)
- average queue length
- response time



The Poisson assumption

Arrival/service time distribution.



$$P(n) = \frac{(\lambda \Delta t)^n e^{-\lambda \Delta t}}{n!}$$

$\langle \Delta t \rangle = \lambda$ $\quad \sigma = \lambda$

Poisson - easy to analyse! ("M")
- limiting case of memoryless for long times.

Simples queue M/M/1 (12.7)

λ = arrival average (per second)
μ = av proces. rate.

dimensionless "traffic intensity" $= \frac{\lambda}{\mu}$

Suppose there are $n-1$ tasks in the queue. To keep the queue length stable, (just cope with demand) we need (on average)

expected arrival rate $(n-1)$ = expected service rate (n)

$$\lambda p_{n-1} = \mu p_n$$

i.e. $\boxed{p_n = \rho\, p_{n-1}}$ (rate eqn) (*)

If $\rho > 1$, clearly n will grow
$\rho < 1$, it could shrink. (let's see!)

i.e. the queue is unstable around $\rho = 1$.

We can solve (*)

$$p_1 = \rho\, p_0$$
$$p_2 = \rho\, p_1 = \rho^2 p_0$$
$$p_n = \rho\, p_{n-1} = \rho^n p_0.$$

10) Queues & Arrivals (chap 12)

Now we assume no error correction needed

Queues occur in all services,

- Random arrivals (average rate)
- Fixed rate of handling / processing.

Queues are classified in Kendall notation. A / S / c

A = arrival distribution (process).

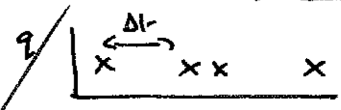
S = service processing distribution

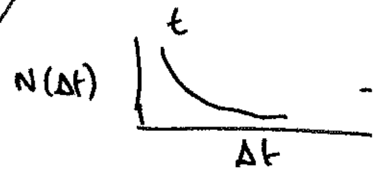
c = number of servers.

Look for average queue lengths.

The Poisson assumption:

Distribution of inter-arrival times





- assumed exponential.

2 reasons: Poisson is the easiest to analyze.

Poisson is a limiting case of independent arrivals (memoryless).

Simple queue M/M/1 (12.7)

M = memoryless, one server.

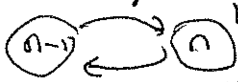
Simple rate balancing.

λ = arrival rate (constant)

μ = service rate

Suppose there are ~~n-1~~ tasks/requests in a queue, then when n have arrived, we had better do some work μ.

$\lambda p_{n-1} = \mu p_n$



Likewise.

$\boxed{p_n = \rho\, p_{n-1}}$ (*) $\rho = \lambda/\mu$

ρ is called the traffic intensity.

If $\rho > 1$, then clearly the queue will probably grow out of control.

If $\rho < 1$, it will tend to shrink.

The queue is unstable.

We can solve (*) the rate equation.

$p_1 = \rho\, p_0$

$p_2 = \rho\, p_1$

$\cdots \boxed{p_n = \rho^n p_0}$

Normalize.

$$\sum_i^{\infty} p_i = 1 = \sum_{i=0}^{\infty} \rho^i p_0$$

geom.

$$= \frac{p_0}{1-\rho} = 1$$

$$\Rightarrow p_0 = (1-\rho)$$

$$\Rightarrow \boxed{p_n = (1-\rho)\rho^n}$$

Note - we allow n to be ∞, so queue length is unrestricted.

Only queue model for which we can compute average length as a fn of ρ.

Expected length

$$\langle n \rangle = \sum_{n=0}^{\infty} p_n n$$

$$= \sum_{n=0}^{\infty} (1-\rho)\rho^n n.$$

$$= \sum_{n=0}^{\infty} \rho^n n - \sum_{n=0}^{\infty} \rho^{n+1} n$$

relabel $n \to n+1$ second term.

$$= \sum_{n=1}^{\infty} \rho^n n - \sum_{n=1}^{\infty} \rho^n (n-1)$$

$$\langle n \rangle = \sum_{n=1}^{\infty} \rho^n = \frac{a}{1-r} = \frac{\rho}{1-\rho}$$

Response (service) time

"~~Little's law~~"

$\lambda = \frac{\text{arrivals}}{\text{time}}$

$\mu = \frac{\text{responses}}{\text{tot time}}$

$\text{utilisation} = \frac{\text{busy time}}{\text{total}}$.

$\text{mean response time} = \frac{\text{busy time}}{\text{no. of responses}}$

Little's law:

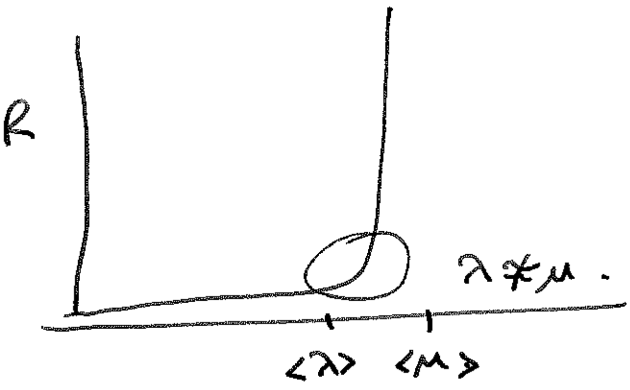
$$\langle \text{jobs} \rangle = \langle \text{arrival rate} \rangle \langle \overset{\text{response}}{\cancel{\text{completion}}} \text{ time} \rangle$$

$$= \left\langle \frac{\overset{\text{response}}{\cancel{\text{arrival}}}}{\text{time}} \right\rangle \left\langle \frac{\text{time}}{\text{response}} \right\rangle$$

$$\langle n \rangle = \mu R$$

$$R = \frac{\langle n \rangle}{\mu} = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu - \lambda}$$

Plot this!



Note however that $\langle \mu \rangle$ is average:

When the server starts to struggle

$\mu_{real} \gg \langle \mu \rangle$

(see gard's paper).

HURST EXPONENT – other distributions

Next week

- more servers, networks

  m/m/k.

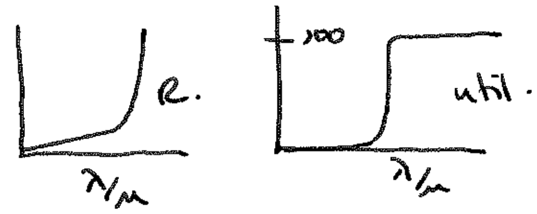
  Load sharing.

11) Workflow + Scalability (ch18)

Last week we showed that simple assumptions allow us to predict that queues will have 2 regimes of behaviour empty or full (power law).

- Queues are more sensitive to arrivals than a deterministic view ~~pred~~ suggests
- danger area $\frac{\lambda}{\mu} = \rho \to 1$. (80/20?)

Queue utilization $= \frac{\text{busy time}}{\text{total time}}$



Load sharing

How can we prevent saturation? Add more servers (k).

$\lambda \to \lambda/k$ for each server

But we have added a bottleneck "the dispatcher" - which is another queue!

More theory:

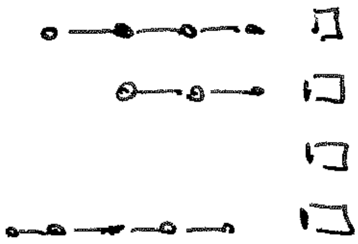
M/M/k queue (server chooses).

behaviour very similar to M/M/1.

Does not account for dispatcher (assumes infinite capacity).

Formulae for $\langle n \rangle$ and R in book. (12-9)

(k - M/M/1) queues $= (M/M/1)^k$



(presorted - arrival chooses)

(even on average, this is not optimal)

Utilization is not exploited on all servers.

Theorem

$\text{Perf}(M/M/k) \geq \text{Perf}(M/M/1)^k$.

Push scheduling $(M/M/1)^k$.

RR - round robin.
LC - least connections etc

Pull scheduling (M/M/k)

Call when ready!

~~Queueing Networks~~ Scaling.

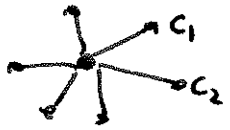
Scaling means: How does a "key performance indicator" change as a function of the size of the system. e.g. N hosts.

Imagine transmission of "jobs"

= work flow

Every network is built from 2 basic patterns:

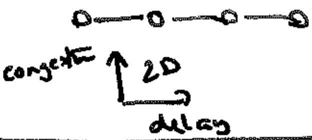
Star/hub



Limited by hub capacity: C

$$C_i = \frac{C_{hub}}{N}$$

chain

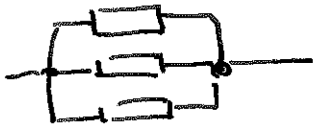


Full capacity at each node, but many delays

$R = R_1 + R_2 + R_3 \ldots$

Like circuit theory

Series + parallel queues (resistors)



More current

$V = IR$, V ↔ demand / pressure intensity

I ↔ work flow

R ↔ server capacity.

Reliability = utilization of time!

$$\text{Availability / Reliability} = \frac{\text{mean uptime}}{\text{total time}} = \rho$$

2 state model:

$$\rho = \frac{MTBF}{MTBF + MTTR}$$

| YES | NO - - - | YES | NO · — — |

Ad-hoc networks

Hosts are not always available for environmental reasons. In a promise viewpoint, the reason is not so important. The effect is the same.

Recall connectivity χ: We can now look at the average or effective connectivity

p_i = prob. host is available.

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & p_1 p_2 & p_1 p_3 \\ & 0 & p_2 p_3 \\ & & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$p_i p_j = p_i$ AND $p_j \Leftrightarrow \text{link}(i,j)$.

Concl.

Probability is the key to understanding

- queues
- scalability.

11) Workflow and Scalability (ch 18)

We have identified work with flow of symbolic information). It is a probabilistic process, subject to faults + errors.

- Need maintenance / repair
- Queueing of services

Our model of workflow is analogous to Ohm's law of electrical circuits, or Kirchoff's laws (for Poisson arrivals)

$$\underbrace{\langle \Delta Q \rangle}_{\text{av. change in state/config}} = \underbrace{I}_{\text{work current}} \underbrace{\Delta t}_{\text{time}}$$

trade off
- capacity
- reliability

$$I = \text{rate of service} = \frac{\text{pressure / demand}}{\text{failures}} = \frac{V}{R}$$

The current is probabilistic $I = I_{max}\, p$ where $(1-p)$ = prob. of failure.

(show if $R = R_0 + R_1(1-p)$, then $I \approx p I_{max}$)

Reliability

$$p = \text{reliability} = \frac{\text{mean uptime}}{\text{total time}}.$$

(time-utilization).

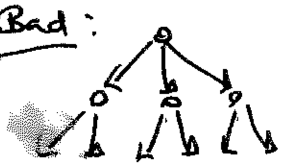
$$p = \frac{MTBF}{MTBF + MTTR}.$$

Failure modes

How do systems delivering services fail?

- Component failures (redundancy)
- Link failures (fault tolerant routes) grids.
- Fault tree analysis (power, net, cooling ...)
- Recall the redundancy folk theorems

Bad:



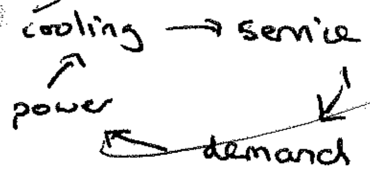
multiple points of failure.

(e.g. load balancing)

The Ohm's law analogy is appropriate because work has to be paid for with electrical power! This generates heat + limits workflow by cooling.

$P \sim IV$

Workflow dependencies:



Unreliable and ad-hoc networks.

Theorem 18.5.1 – a fixed network with unreliable components is equivalent to an ad hoc network of reliable components on average.

- Recall connectivity χ (lecture 7)

$$\langle \chi \rangle = \frac{1}{N(N-1)} \vec{h}^T \langle A \rangle \vec{h}$$

e.g. availability for service

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & p_1 p_2 & p_1 p_3 \\ & 0 & p_2 p_3 \\ & & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

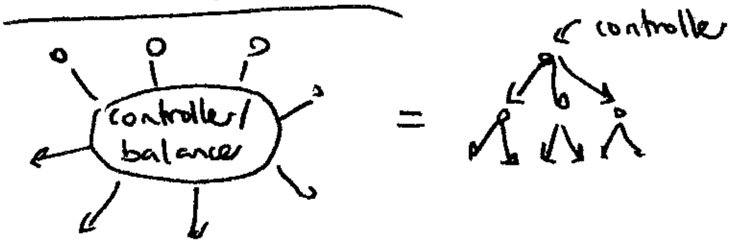
$p_i p_j = p_i$ AND $p_j \Leftrightarrow \text{link}(i,j)$.

Flow Architectures

rate error vs rate correction

There are 6 levels of architecture from a centralized (star / tree) model to peer 2 peer. (increasing amounts of redundancy or fault tolerance. See 18.6)

Model 1: star/ tree



I (controller) $= I_1 + I_2 + \cdots I_N \cdots$

$\langle I_{load} \rangle = N I_{component}$

or $I_{comp} = \dfrac{I_{load}}{N}$ (bottleneck).

Ad hoc $\in I_{load} \propto \sqrt{N} \sim \cancel{\frac{I_0}{\sqrt{N}}}$

- If N is large, too much for controller to do.
- If N is small components get a greater share of work.

Mode 2 - unreliability does not change the average scaling from the architecture viewpoint.

Models ... 6

Increasing autonomy, independent of N.

Centralized models are strongly dependent on "N" - number of nodes being shared. Cannot always avoid this (routing - one route to net? power supply - one source?

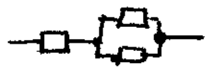
Peer models - harder to understand, hot research topic.

Scalability - where are resources / clients ⇒ topology!

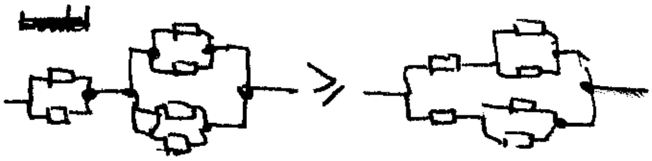
Server room example

- Redundant everything!
  - net, ISP
  - power supply
  - cooling.
  - components.

Recall low level redundancy theorem.

Let S_1 = 

~~bundle~~



Queuing Networks

Parallel - M/M/k queue analysis.

Series - Response time law p.185 applies to M/M/k queues. in series.

Can find out where bottlenecks are from response times. G/G/k queues are much harder.

Last

Queues - Arrivals / completions M/M/1 → k.
~~random process~~
Utilization law
Flow analogy.

Clearly as $\rho \to 1$, $\langle n \rangle \to \infty$

Some basic laws:

$\lambda = \dfrac{\text{Arrivals}}{\text{time}}$

$\mu = \dfrac{\text{Completions}}{\text{time}}$ (throughput)

$\text{Utilization} = \dfrac{\text{Busy time}}{\text{total time}} = U$

Mean service time = Busy / completions $= B/C$

Utilization law

$$U = \frac{B}{T} = \frac{C}{T} \times \frac{B}{C}$$

$$U = \mu S$$

obs back tsp?

(~~ave~~ throughput) $\times$ (mean service time)

Recall flow results (isomorphic).

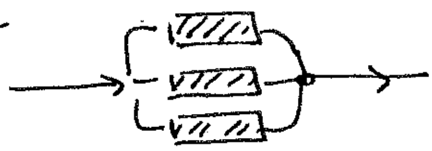
Ohm's law: $V = IR$.

$U = \mu S$

Serial and parallel queues.

(Like resistors in series) Service time adds $S_{serial} = \sum S_i$.

or



First queue to respond allows through.

$$\frac{1}{S_{parallel}} = \frac{1}{S_1} + \frac{1}{S_2} + \dots$$

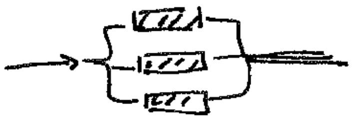
Folk theorems (18.2)

A parallel coupling of queues is never worse than a better component

A serial coupling of components is never better than its weakest link.

M/M/k model

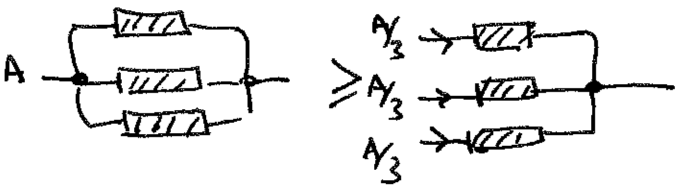
We can ~~always~~ also solve the case of



k servers.

The results:

One queue and k servers is better than k independent queues. i.e.



This wastes time if one queue is idle.

Hurst Exponent (10.10)

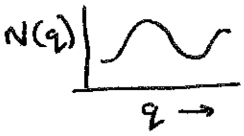
Coarse graining at different scales. If fluctuations have only one scale, expect smoothing.

$$s^{-H} q(st) = q(t)$$

9) Information & integrity (ch. 9) 15

A model of comm. or effective work.

Claude Shannon ++, a model of statistical properties of distributions.
(change = information)



Applies to frequency of observed values q.

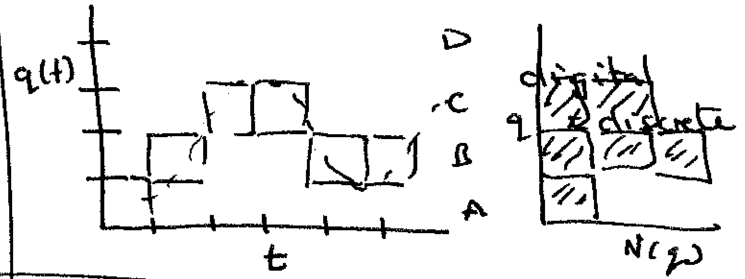
~~Distributions q occur in~~

- ~~categorization / classification~~
- ~~symbolism A/D, D/A conversion~~
- ~~variation of "signals" or "messages" (config)~~

Statistics of dist's:

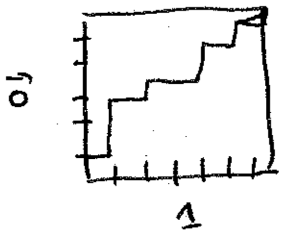
- distribution of resources (optimization)
- randomness (property of distributions) variation over time
- communication / noise (discrete symbols)

The essence lies in this picture:



Discrete = digital
continuous = "analogue".

Information is slightly counter-intuitive.
Consider a binary message of length N.
We can draw this on a grid.



Total no. of paths

$$h = \frac{(n_0+n_1)!}{n_0!\,n_1!}$$

$N = n_0 + n_1$

Define $\log h / N \equiv \underline{\underline{H}}$ info per symbol.

Think of h as the 'hopelessness' of finding one path amongst all possible (uncertainty).

Paradoxically, also tells us how much information we must specify to ~~describe~~ distinguish this one path from others.

H = entropy = uncertainty = information

Can show: (9.8)

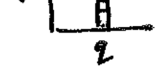
$$\boxed{H = -\sum_i^{C} p_i \log_m p_i}$$

where alphabet $\Sigma = \{ s_1, s_2 \dots s_c \}$

- Notice p_i = probability of symbol i occurring. $p_i = n_i / N$.
- No memory of the actual sequence, only statistical dist.

Entropy is ~~maximum~~ min if $p_{i=\alpha} = 1$.

i.e.



$\Rightarrow H = p_\alpha \log_m p_\alpha = \underline{\underline{0}}$

Entropy is ~~minimum~~ max if $p_i = \frac{1}{C}, \forall i$.

$$H_{min} = -\sum^{C} \frac{1}{C} \log_m \frac{1}{C}$$

$$= \underline{\log_m C}.$$

If m=2, H is measured in bits per symbol

- H provides a theoretical lower limit on the average length (in compression) for a message to be transmitted. (e.g. 9.9)

- Entropy tells us how "flat" or "distributed" a dist is.

Application (9.11)

Find the best spread / most distrib. configuration of categories $i \in \Sigma$.

- Maximise H as fn. of p_i, with additional constraints.
- Use Lagrange metode.

$$L = H - \alpha\left(\sum p_i - 1\right) - \beta\left(f(p) - \chi\right)$$

where $f(p) = \chi$ is a constraint.

$$\frac{\partial L}{\partial p_i} = 0, \quad \frac{\partial L}{\partial \alpha} = 0, \quad \frac{\partial L}{\partial \beta} = 0.$$

$$\boxed{H(I;O) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{p_i p_j}} \quad (9.22)$$

'Knowledge' - expert

We believe we 'know' something if we know how to put it into the right box. ie. identifying the correct symbol to label something. Entropy tells us about this.

Information in a signal. (15.6)

Mutual information transfer (9.7)

Transmit symbols from $\Sigma = \{\ldots\}$ from input to output. $I \longrightarrow O$

Joint probability $p(I,O)$ = matrix of probability that if I at input, we get O at output. $I, O \in \Sigma$

The information transmitted, over time, in ~~bits~~ "symbols, per symbol."

$m = 2$ = bits per symbol. $= H(I;O)$

Tells us about ~~[illegible]~~. 

Continuum of symbols, Gaussian distribution of $p_i \to p(x)$. (noise). (see 15.6)

$$p(q) \sim e^{-q^2/\sigma^2}$$

$$H(q) = -\sum p(q) \log_2 p(q)$$

constraint

$$\langle q^2 \rangle = \frac{1}{\Delta T} \int_0^{\Delta T} q^2(t)\, dt = \underline{\underline{P}} \quad \text{(power)}$$

$$\boxed{H = \frac{1}{2} \log_2\left(1 + S/N\right)} = \text{channel capacity per bit stream}$$

$$q(t) = S + N = \langle q \rangle + \delta q. \quad \text{(lecture 2)}$$

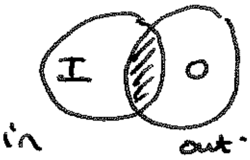
So the twice sampled channel capacity

$$C = 2BH$$

2

How "flat" is the distribution?

Mutual information (copying).



Transmit a string of symbols

Av information in input AND output

$$\boxed{H(I;O) = \sum_{ij} P_{ij} \log \frac{P_{ij}}{P_i P_j}}$$

P_{ij} = adjacency matrix for I/O symbols.

Gaussian Signals (9.11)

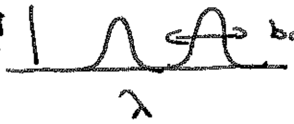
Find distribution of input which maximizes entropy, if average distance from average = σ.

$L = H - \alpha(\sum p_i - 1) - \beta(\chi)$.

$\Rightarrow \; p_i = e^{-}$

(Ex 43).

Signals:



$p(q) = e^{-q^2/\sigma^2}$

$\langle q^2 \rangle = \frac{1}{\Delta T}\int_0^{\Delta T} q^2\, dt = P$ (power).

~~Maximize~~ Find $\overline{S+N = P}$ $H = -\int p \log p \, dq$

$$\boxed{H = \frac{1}{2}\log_2\left(1 + \frac{S}{N}\right).}$$

True for each "band"

= channel capacity = bits per symbol for one band. alphabet.

~~[illegible]~~

$C = \underbrace{2B}_{} \times H.$

~~{1,0}~~ ~~symbols~~ ~~freq Hz~~. samples (per sec). Nyquist.

bits / ~~symbol~~. sample

C = av bits / ~~[illegible]~~ / sec.

(without redundancy) upper limit

Conclusion

We've now related symbols / change operations / information

idempotence $\Leftrightarrow$ compression. convergence

channel capacity $\Leftrightarrow$ Gaussian signals

Error correction $\Leftrightarrow$ mutual info?

Our story about change is finished. Things we take for granted (error correction) have been explained.

We understand "maintenance" "scheduling"

Next: how to use this to make judgements?

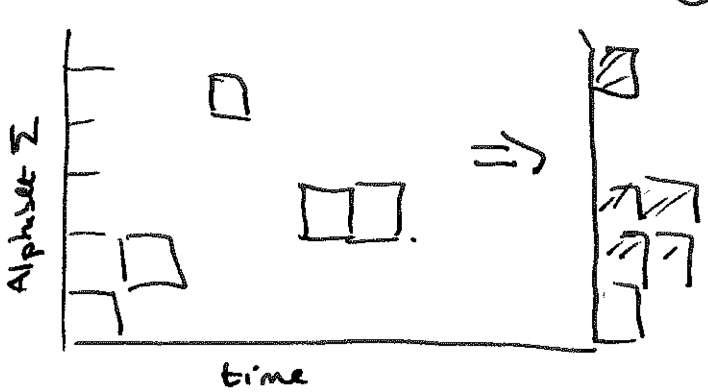
9) Information + Integrity · ch 9, 15

Take an alphabet Σ and form strings of symbols $\Rightarrow$ information.

- How much?
- Is it unique (non-compressible?) (idempotence)

Look for amount of incompressible information per symbol.

(This is a measure of "convergent configuration").



- Turn a time-series into a histogram.
- Look at the statistics of the signal.

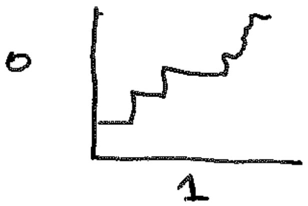
Use this to discuss:

1. distribution (even) of resources
2. amount of wasted effort (redundant symbols)

Av Information content per symbol.

Consider a binary message $\Sigma = \{1, 0\}$ of length N. $N(1)$ = no. of 1's etc.

Can draw this as a path in Σ space:



Total no. of unique messages we can make = total no. of paths.

$$h = \frac{(\text{no. of perms of message})}{(\text{redundancy of 0's})(\text{red. of 1's})}$$

$$= \frac{(N(1) + N(0))!}{N(1)!\,(N(0))!}$$

$N = N(0) + N(1)$. (hopelessness)

$\text{Log}_2(\text{number})$ = no. of bits to code "number".

e.g. 4 bits (0–31), $2^4 = 32$. etc.

Define

$$\frac{\text{Av. bits of label}}{\text{input symbols}} = \frac{\log h}{N}$$

see (9.8) def $p_i = \frac{n_i}{N}$

$$\boxed{H = -\sum_i^{c} p_i \log_2 p_i}$$

- Entirely av result - Does not "remember" content of message.
- Provides a theoretical limit on compressability of message.

H = Shannon entropy.

Minimum entropy (least info. per symbol)

$$H_{min} = -\sum_i^{c} \frac{1}{c} \log_m \frac{1}{c} = \log_m c.$$

e.g. $\log_2 2 = 1$. bits per symbol.

Max ent

when $p_{i=\alpha} = 1$ $p_{i\neq\alpha} = 0$

$H_{max} = 1 \log 1 = 0$

12) Decision-making (ch 19)

- How to make rational decisions by formalizing questions + process.
- Optimization of goals described by policy. (steady state).
- Note this is relativistic, i.e. best is not alway decidable without arbitrary choices (policy).

i) Decisions require us to categorize or classify our options.

- think of these as symbols in an alphabet Σ of operations / chgs

ii) Decisions sometimes involve conflicting interests.

- how do different goals compete?
- how do we resolve "conflicts" rationally? (alternatives)

iii) Discrete, deterministic decisions

- Boolean algebra, AND/OR/NOT

(you know this)

iv) Continuous, non-deterministic decisions (mixtures) ~~note~~ "tuning"

- value / currency based
- payoff / value / utility / outcome
- algebra for optimizing result

Valuation
~~Causality / Multiple possibilities~~

~~If we have~~ Weight the importance of different choices by the value of their expected payoff.

Do we pick only one strategy or a mixture of several to maximize gain?

Theory of Games is about competition in terms of cost. between different goals in a common 'space'. Goals = "players", players = 2 here

Two kinds of game-model:

i) Discrete, extensive form.

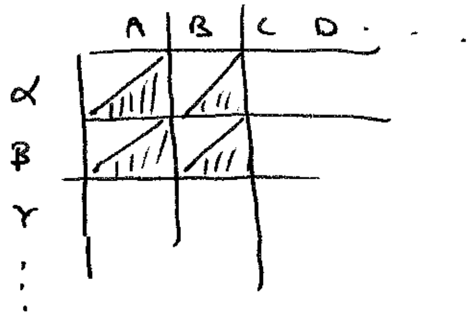
- Make a tree of every possible action from every player. (ontology)
- very complex
- micromanagement
- add up payoffs and maximize
- player 1, then player 2, then player 1, ...

ii) Continuum (averaged), strategic form

- also called normal form.
- lump together all ~~each~~ actions into average strategies that characterize different classes of play.
- high level management
- player 1, player 2 viewed as "acting together".

equilibria. e.g., principal agent

We shall focus on high-level decisions with the normal form.



Payoff matrix Π_{ij}

A, B, C.. = player 1's strategies (pure)

α, β, γ.. player 2's strategies

Players can also use <u>mixed</u> strategies

or $c_1 A + c_2 B + c_3 C + \cdots$

where this is mixed during a single game or over several games.

e.g. upgrading software game.

Let payoff Π = convenience to <u>users</u>

Players = { 'forces of evil' + users ; system administrator }

	security hole	other bug	missing function
upgrade	10, 5	10, 0	5, −5
test first	5, 5	3, 9	0, 8
keep multiple versions	−10, 5	−1, 10	0, 10

(sysadm, users) really <u>two</u> matrices.

We see that payoff can be quite subjective.

- Does it matter, as long as consistent?

<u>Making the decision</u>

- A 'solution' of the game is a specification of optimal strategies for each player, and the value received by each player.
- Both players try to maximize their payoff simultaneously.

<u>Methods</u>

i) if $\Pi_1 + \Pi_2 = 0$ (zero-sum game)

the game is solved by <u>minimax</u> or mixed strategy minimax.

ii) if $\Pi_1 + \Pi_2 \neq [\text{const}]$ can use equilibrium ideas, e.g. Nash equilibrium.

Next week: solutions and examples.