Modeling Telecommunications Traffic

Measurements and models

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

"Measure what is measurable, and make measurable what is not so."

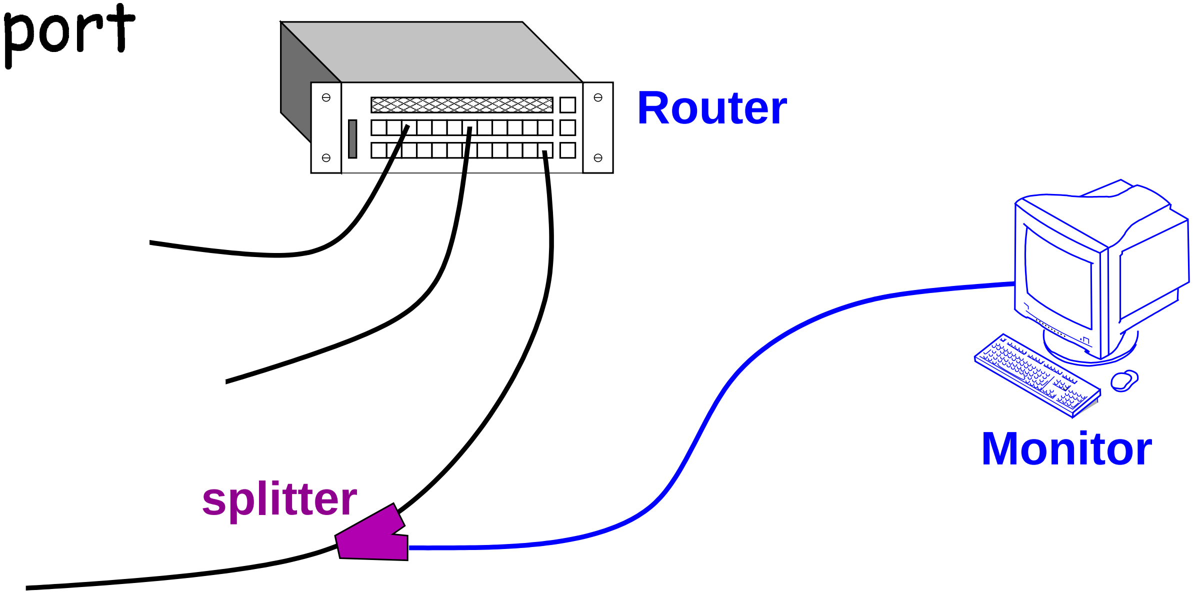
Galilei, Galileo (1564 - 1642)

Outline

- Internet traffic measurements
  - Packet traces
  - Netflow
  - Sampled netflow
  - SNMP
- Measurement based models
  - Structural models
  - On/Off processes
  - Doubly Stochastic Poisson Process
  - Generalizations
  - Aggregate traffic models

Packet traces

- tap a link, or router
  - optical, or electronic splitter/coupler
  - monitoring port



- record every packet's
  - size
  - time (of first byte)
  - headers (IP, TCP, possibly more)

Packet traces issues

- timing resolution/accuracy
  - clock resolution (packet transmission time for 1500 byte packet on OC48, 2.5 Gbps, is 4.8 microseconds)
  - clock accuracy: PC clocks have drift, plus interupt latency
  - 2.5 Gbps, min size (40 byte) packets, you have 128 ns to timestamp the packet
- storing the data
  - OC48 2.5 Gbps data rate
  - minimum size packets are basically all header
  - need to get 2.5 Gbps to disk (which is hard)

Packet trace info

- IP header
  - version, header length, TTL, checksum
  - flags: ToS, ...
  - packet length (size in octets/bytes)
  - source and destination address
  - options
- TCP/UDP header
  - source and destination ports
  - sequence, and ACK numbers, checksum
  - flags: SYN, ACK, ...
  - data offset and pointers
  - options

tcpdump output

```
1078208222.013538 129.127.5.110.1346 > 229.55.150.208.1345: udp 150
1078208222.754748 129.127.5.117.631 > 129.127.5.255.631: udp 139
1078208222.948664 129.127.5.56.1025 > 129.127.5.255.111: udp 136
1078208222.948673 129.127.5.56.1025 > 224.0.2.2.111: udp 136
1078208223.257521 129.127.5.234.1346 > 229.55.150.208.1345: udp 150
1078208223.516606 129.127.4.9.513 > 129.127.5.255.513: udp 108 (DF) [ttl
1078208223.755331 129.127.5.117.631 > 129.127.5.255.631: udp 137
1078208224.755755 129.127.5.117.631 > 129.127.5.255.631: udp 133
1078208225.756207 129.127.5.117.631 > 129.127.5.255.631: udp 158
1078208228.137869 129.127.5.56.1025 > 129.127.5.255.111: udp 136
1078208228.137881 129.127.5.56.1025 > 224.0.2.2.111: udp 136
1078208231.728471 129.127.4.177.5353 > 224.0.0.251.5353: udp 105
1078208233.257055 129.127.5.56.1025 > 129.127.5.255.111: udp 136
1078208233.257066 129.127.5.56.1025 > 224.0.2.2.111: udp 136
```

Packet trace example

Packet trace (snippet)

	IP header				TCP/UDP header	
timestamp	proto.	src IP	dst IP	size	src port	dst port
1078208222.014	udp	129.127.5.110	229.55.150.208	150	1346	1345
1078208222.755	udp	129.127.5.117	129.127.5.255	139	631	631
1078208222.949	udp	129.127.5.56	129.127.5.255	136	1025	111
1078208222.949	udp	129.127.5.56	224.0.2.2	136	1025	111
1078208223.258	udp	129.127.5.234	229.55.150.208	150	1346	1345
1078208223.517	udp	129.127.4.9	129.127.5.255	108	513	513
1078208223.755	udp	129.127.5.117	129.127.5.255	137	631	631
1078208224.756	udp	129.127.5.117	129.127.5.255	133	631	631
1078208225.756	udp	129.127.5.117	129.127.5.255	158	631	631
1078208228.138	udp	129.127.5.56	129.127.5.255	136	1025	111
1078208228.138	udp	129.127.5.56	224.0.2.2	136	1025	111
1078208231.728	udp	129.127.4.177	224.0.0.251	105	5353	5353
1078208233.257	udp	129.127.5.56	129.127.5.255	136	1025	111
1078208233.257	udp	129.127.5.56	224.0.2.2	136	1025	111

Packet traces pros

- get to see almost everything
  - source
  - destination
  - ports and protocol
  - TCP flags
- to see everything, need to store more than just 40 bytes (e.g. need application headers)
  - but you can!
- very fine grained (timewise)
- suitable for just about any type of modeling

Packet traces cons

- cost of monitors (1 per link)
  - can put multiple cards/ports on one monitor for low speed monitors
  - have to add installation and maintenance costs
- ginormous datasets
  - at OC48, it takes less than 1 hour to collect a terabyte (min sized packets)
  - even with 1500 byte packet, it only takes 33 hours to collect a terabyte.
  - even simple processing is slow!

Reducing the data size

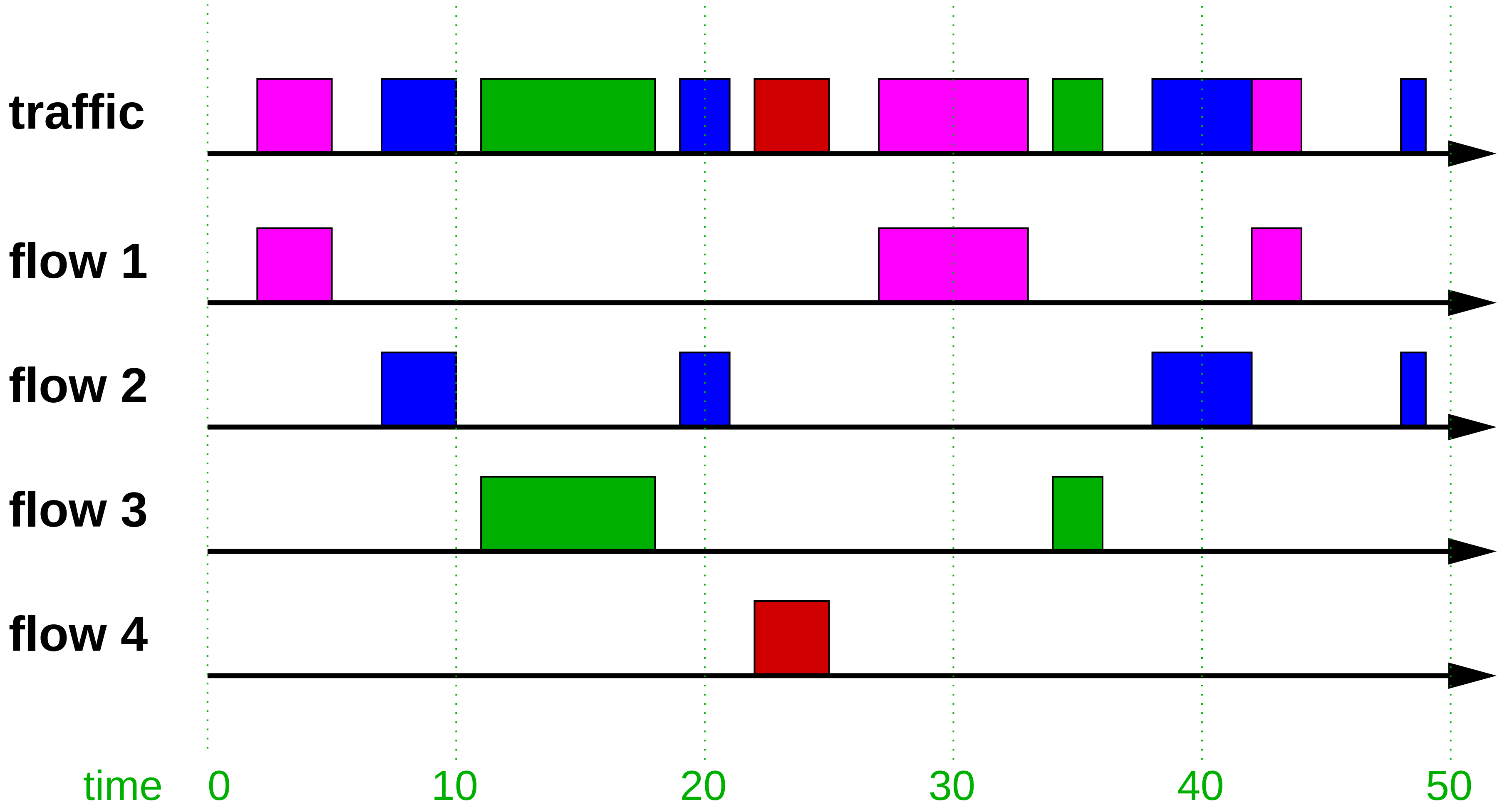
A number of operations can reduce the dataset size

- **sampling:**
  - standard statistical approach
  - simplest case, sample every Nth packet, or randomly choose 1 in N packets.
- **filtering:** only look at packets which meet certain requirements, e.g.
  - only TCP packets
  - only packets between two specific IP addresses
- **aggregation:** reduce the granularity of the data somehow.
  - aggregate over time, or **keys**

Netflow

- idea: aggregate to close to a TCP connection
  - keep one record per flow
  - record key: IP source, dest, protocol and TCP source, dest port
  - record stores: packets, bytes, TCP flags, start and stop time
- practicality: aggregate by key
  - flush records using
    - timeout, O(30 seconds), (to separate similar connections, e.g. DNS)
    - when flow record cache is full
    - every 15 minutes (stop staleness of records)
  - not bi-directional

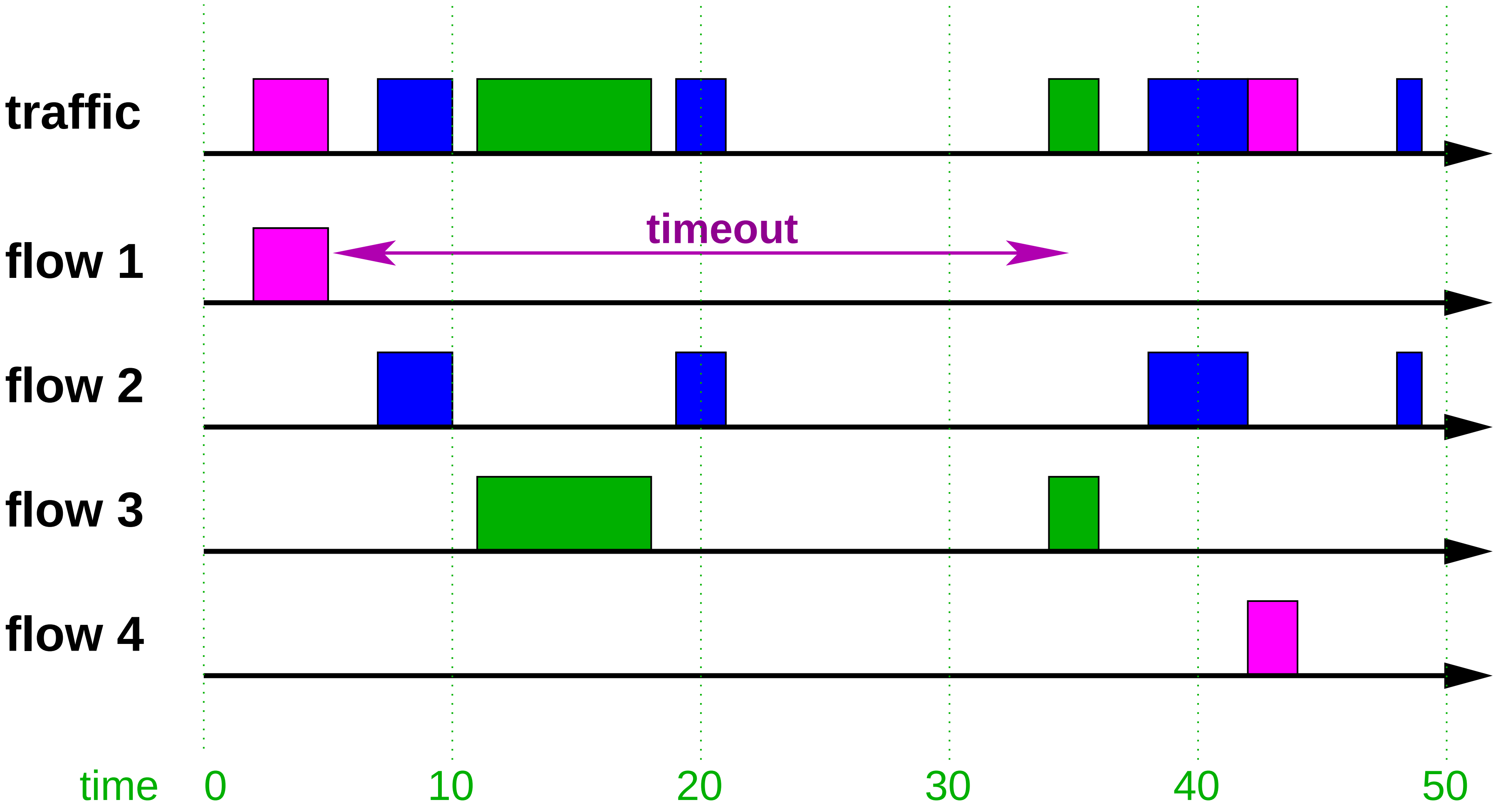
Netflow



Netflow records

key	packets	bytes	start time	stop time
pink	3	11	2	44
blue	4	10	7	49
green	2	9	11	6
red	1	3	22	25

Timeouts



Netflow records

key	packets	bytes	start time	stop time
pink 1	1	3	2	5
blue	4	10	7	49
green	2	9	11	6
pink 2	1	2	42	44

Netflow example

Packet trace (snippet)

timestamp	protocol	src IP	dst IP	src port	dst port	size
1078208222.014	udp	129.127.5.110	229.55.150.208	1346	1345	150
1078208222.755	udp	129.127.5.117	129.127.5.255	631	631	139
1078208222.949	udp	129.127.5.56	129.127.5.255	1025	111	136
1078208222.949	udp	129.127.5.56	224.0.2.2	1025	111	136
1078208223.258	udp	129.127.5.234	229.55.150.208	1346	1345	150
1078208223.517	udp	129.127.4.9	129.127.5.255	513	513	108
1078208223.755	udp	129.127.5.117	129.127.5.255	631	631	137
1078208224.756	udp	129.127.5.117	129.127.5.255	631	631	133
1078208225.756	udp	129.127.5.117	129.127.5.255	631	631	158
1078208228.138	udp	129.127.5.56	129.127.5.255	1025	111	136
1078208228.138	udp	129.127.5.56	224.0.2.2	1025	111	136
1078208231.728	udp	129.127.4.177	224.0.0.251	5353	5353	105
1078208233.257	udp	129.127.5.56	129.127.5.255	1025	111	136
1078208233.257	udp	129.127.5.56	224.0.2.2	1025	111	136

Netflow example

Netflow records:

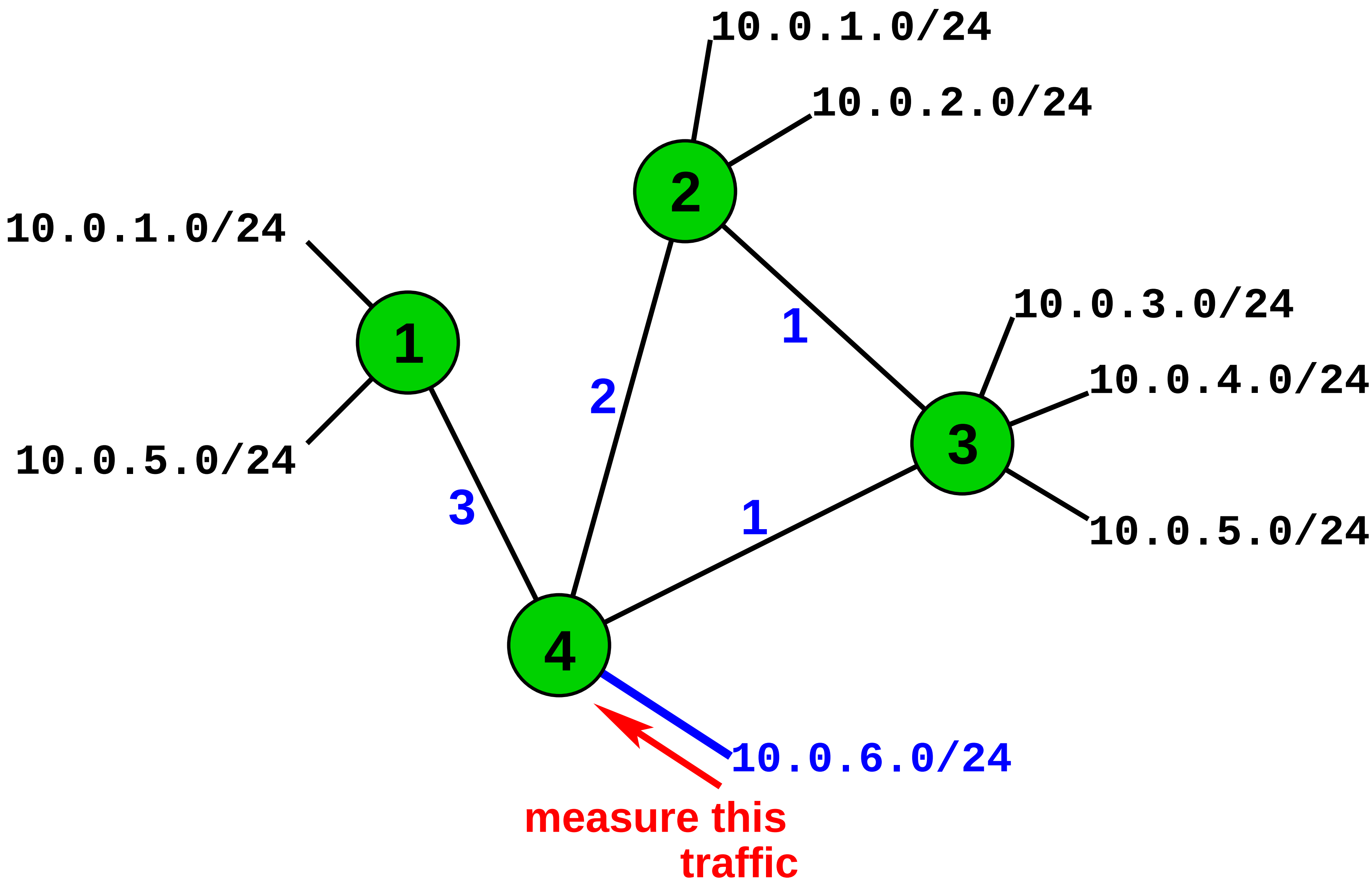
protocol	src IP	dst IP	src port	dst port	dur	pack.	bytes
udp	129.127.5.234	229.55.150.208	1346	1345	0.0	1	154
udp	129.127.5.56	224.0.2.2	1025	111	10.3	3	410
udp	129.127.5.110	229.55.150.208	1346	1345	0.0	1	154
udp	129.127.4.177	224.0.0.251	5353	5353	0.0	1	102
udp	129.127.5.117	129.127.5.255	631	631	3.0	4	563
udp	129.127.5.56	129.127.5.255	1025	111	10.3	3	410
udp	129.127.4.9	129.127.5.255	513	513	0.0	1	113

Netflow example application

Traffic matrix

- measure netflow at network entry points (ingress)
  - provides traffic from IP source to dest. address
- aggregate to prefix level (across all ports)
  - get a matrix from IP source prefix to destination prefix
  - matrix is sparse, but large (100k+ prefixes)
  - also, one matrix per ingress point to the network
- to get ingress/egress traffic matrix, need to
  - simulate routing, to compute egress points per ingress, and prefix
  - then aggregate again

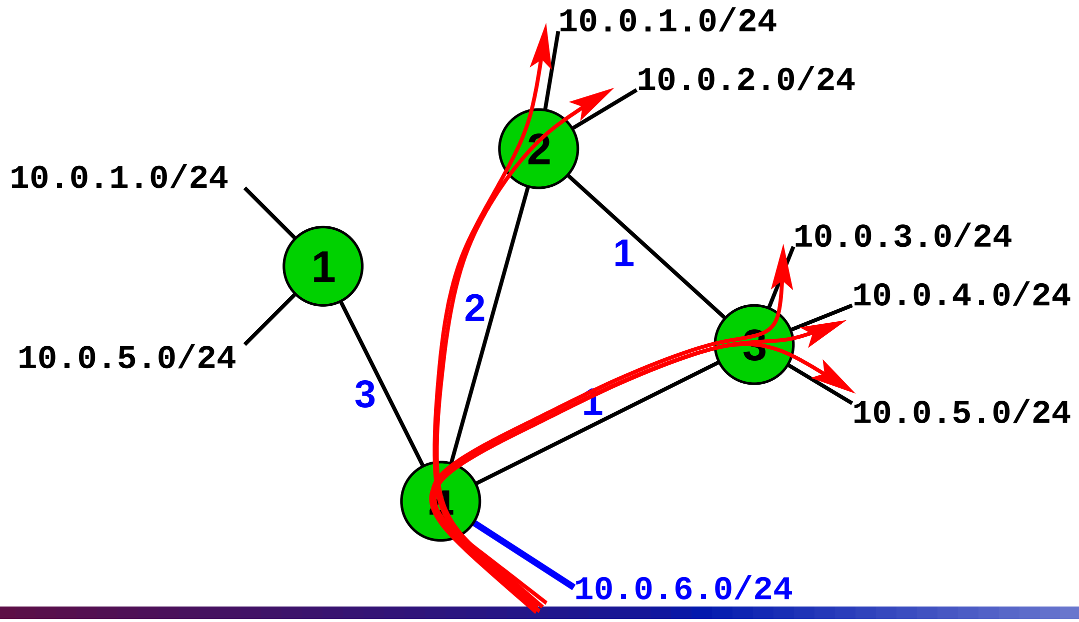
Netflow example application



Example traffic matrix computation

Measured incoming traffic at node 4

ingress node	source prefix	dest prefix	volume	egress node
4	10.0.6.0/24	10.0.1.0/24	10	2
4	10.0.6.0/24	10.0.2.0/24	11	2
4	10.0.6.0/24	10.0.3.0/24	21	3
4	10.0.6.0/24	10.0.4.0/24	6	3
4	10.0.6.0/24	10.0.5.0/24	3	3



Example traffic matrix computation

Ingress-egress traffic

ingress node	egress node	volume
4	1	0
4	2	21
4	3	30
4	4	0

Ingress-egress traffic matrix

		egress node			
		1	2	3	4
ingress node	1	-	-	-	-
	2	-	-	-	-
	3	-	-	-	-
	4	0	21	30	0

Netflow pros

- data volume reduction
  - 100:1 reduction on packet traces
  - conservative estimate for real traffic
- collected by router
  - doesn't require special equipment
  - no maintenance cost
- almost standard these days
- keeps much of the useful traffic parameters
  - flows map (somewhat) to connections
  - can see where traffic is going
  - port numbers still visible

Netflow cons

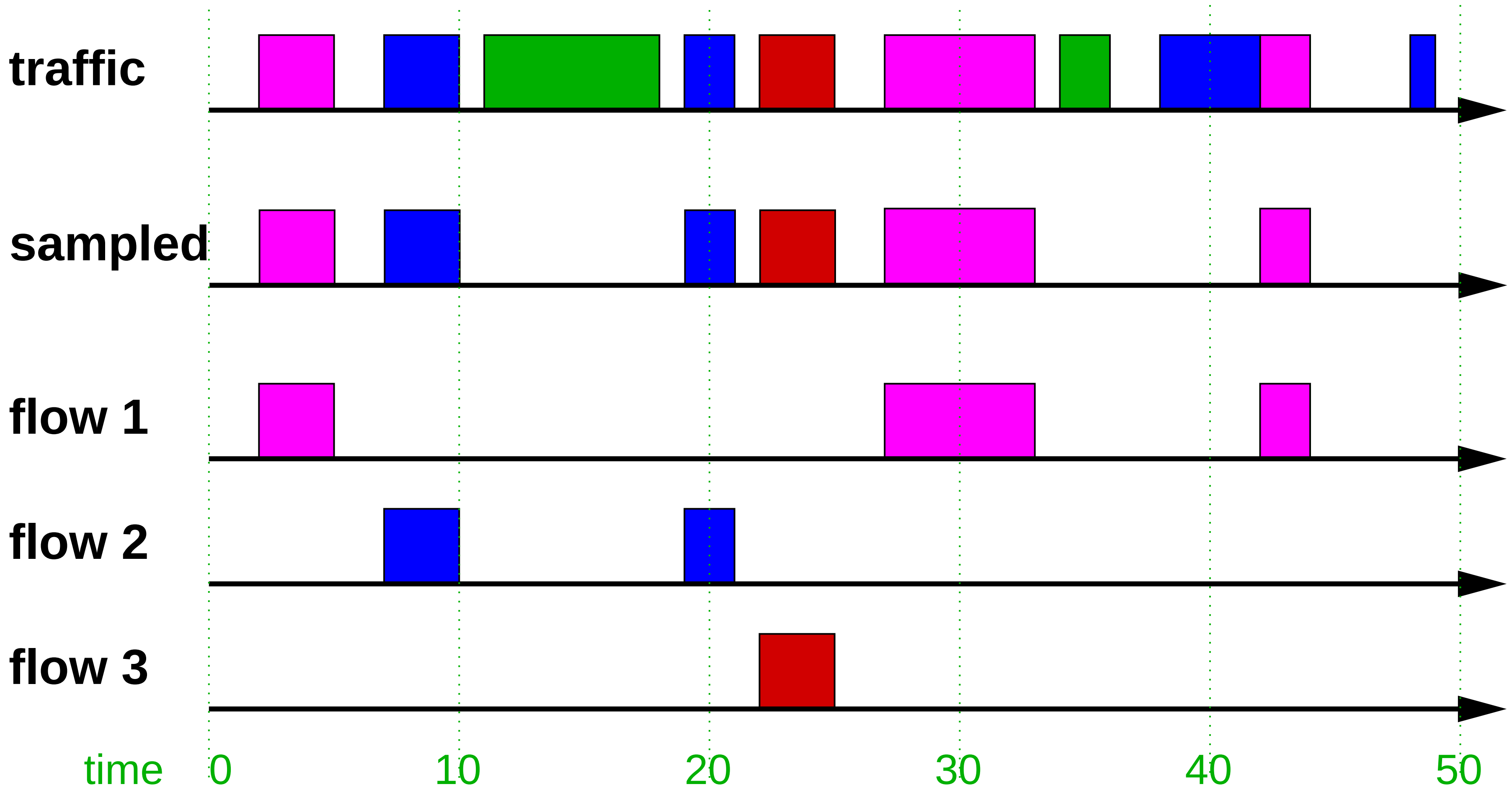
- historically poor vendor support
  - feature interations
  - bugs
  - performance impact on router
- still large volumes
  - may require special equipment for data reduction
- loose some detail
  - application level headers are now lost forever
  - loose time granularity
    - only have start and stop of flows
    - several minutes (cache flushing cap)
    - don't see traffic per time interval

Port based application classification

- ideal
  - particular TCP ports used by particular servers
  - port usage by applications is
  - should be able to classify traffic by TCP ports
- real
  - ports used are often not registered
  - ports ports may be misused
  - same application may have different use cases
  - port based classification doesn't work well

Sampled Netflow

- bit of a misnomer
- really means netflow of sampled packets



Netflow records

Netflow

key	packets	bytes	start time	stop time
pink	3	11	2	44
blue	4	10	7	49
green	2	9	11	6
red	1	3	22	25

Sampled Netflow

key	packets	bytes	start time	stop time
pink	3	11	2	44
blue	2	5	7	21
red	1	3	22	25

Impact of sampling

- main advantage: reduced cost to router
  - less cache memory needed for storing records
  - fewer packets need to be added to flow records
  - smaller amount of data exported
- main disadvantage: distortion of traffic stats
  - negligable impact (under certain assumptions)
    - traffic matrix
  - biases (reversable)
    - flow duration and size distribution
    - longer/bigger flows more likely to be detected
  - biases (unreversable)
    - second order stats (autocorrelation)

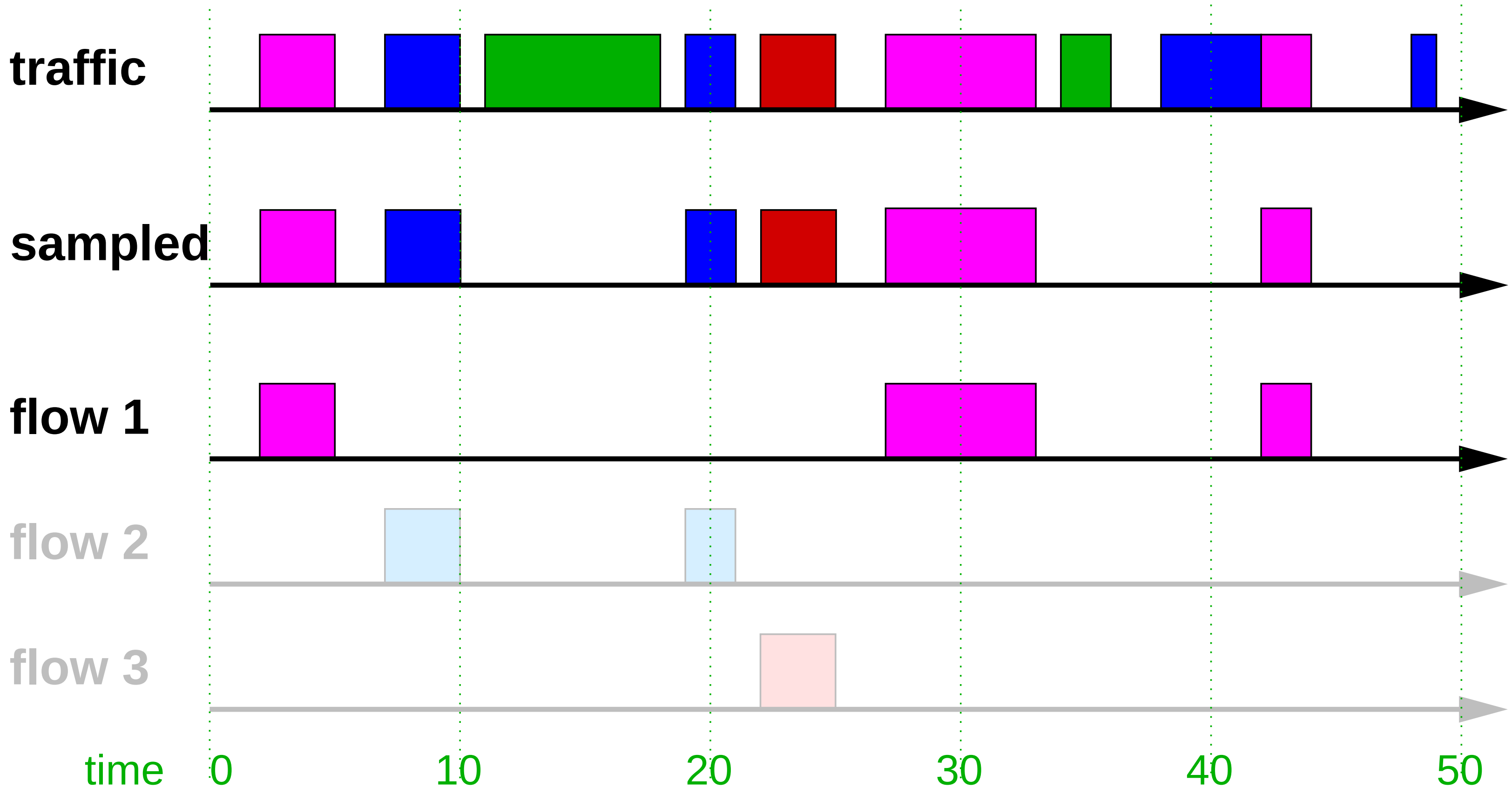
Example distortion

Application distribution example

- network has two applications
  - **application A:** flow size 1000 packets
    10% of traffic
  - **application B:** flow size 1 packet
    90% of traffic
- sample 1 in 100 packets (randomly)
- probability of detecting a flow in the samples
  - **application A:** $p = 1 - (1 - 1/100)^{1000} \simeq 1$
  - **application B:** $p = 1/100$
- relative volume of sampled flows:
  91.7% application A, and 8.3% application B.

Netflow samples

- two layers of sampling
- packet sampling (pre aggregation)
- flow sampling (post aggregation)



Netflow records

Netflow

key	packets	bytes	start time	stop time
pink	3	11	2	44
blue	4	10	7	49
green	2	9	11	6
red	1	3	22	25

Netflow Samples

key	packets	bytes	start time	stop time
pink	3	11	2	44

Impact of sampling (flows)

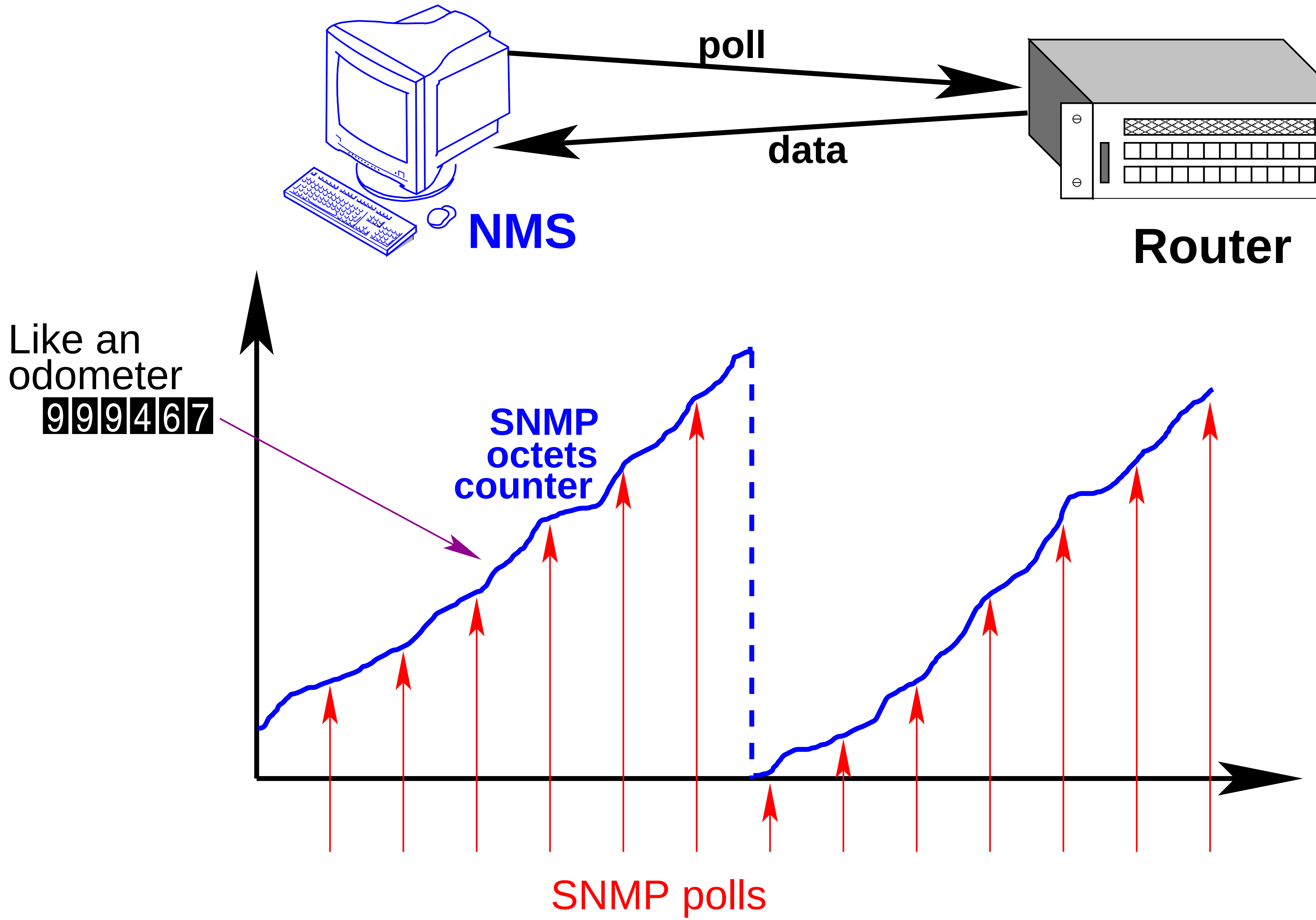
- advantages
  - fewer flow records, so less storage needed
  - little bias introduced
- disadvantages
  - increased variance of traffic stats
    - standard effect of sampling
    - exacerbated because large (heavy-tailed) flows contribute disporportionally to the traffic volume, but occur rarely (so may be lost in sampling)
    - can be controlled by using smart sampling scheme

SNMP

Simple Network Management Protocol

- not just for measurements
- allows one to collect MIBs (Management Information Bases)
- MIB-II implemented on almost all network equipment
- includes:
  - counts of packets
  - counts of bytes
  - ...

SNMP data collection



Irregular sampling

- Why?
  - missing data (transport on UDP, in-band)
  - delays in polls
  - poler sync (mulitple pollers)
  - staggered polls
- Why care?
  - time series analysis
  - comparisons/totals between links
  - correlation to other data sources

SNMP pros

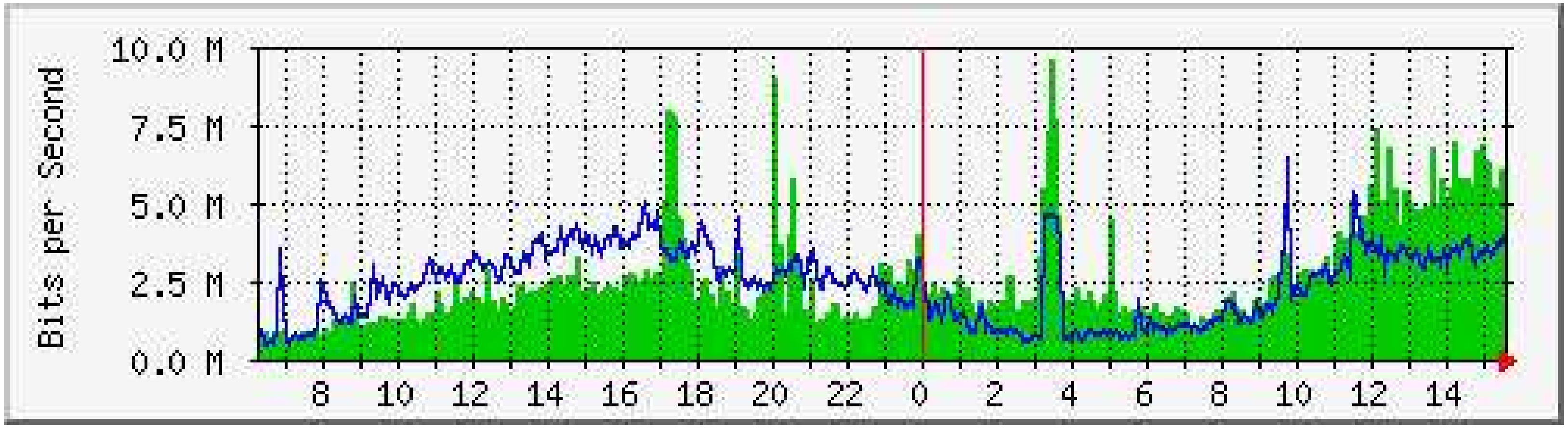
- simple, and easy
- low overhead
- ubiquitous
- lots of practice in use
- lots of historical data

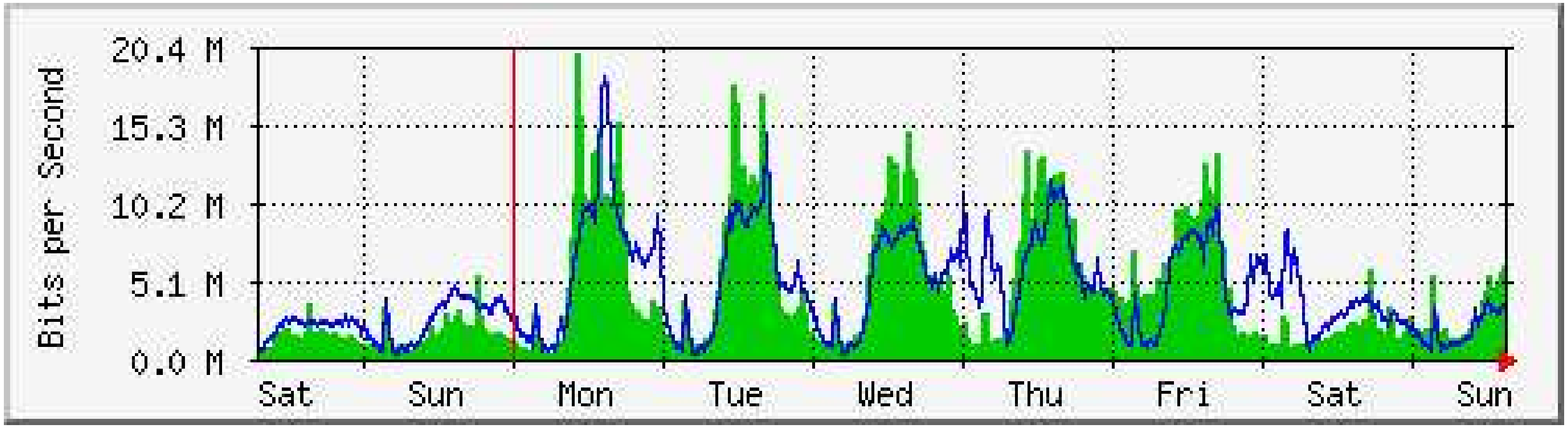
SNMP cons

- data quality
  - missing data
  - ambiguous data
  - irregular time sampling
- coarse time scale (> 1 minutes, typically 5)
- octet counters don't tell you
  - what type of traffic (applications)
  - where traffic is going (source and destination)
  - hard to detect DoS, or other such attacks
- coarse time scale (> 1 minutes, typically 5)

Example SNMP data

RRD Tool (or MRTG) are the commonest form of available SNMP data

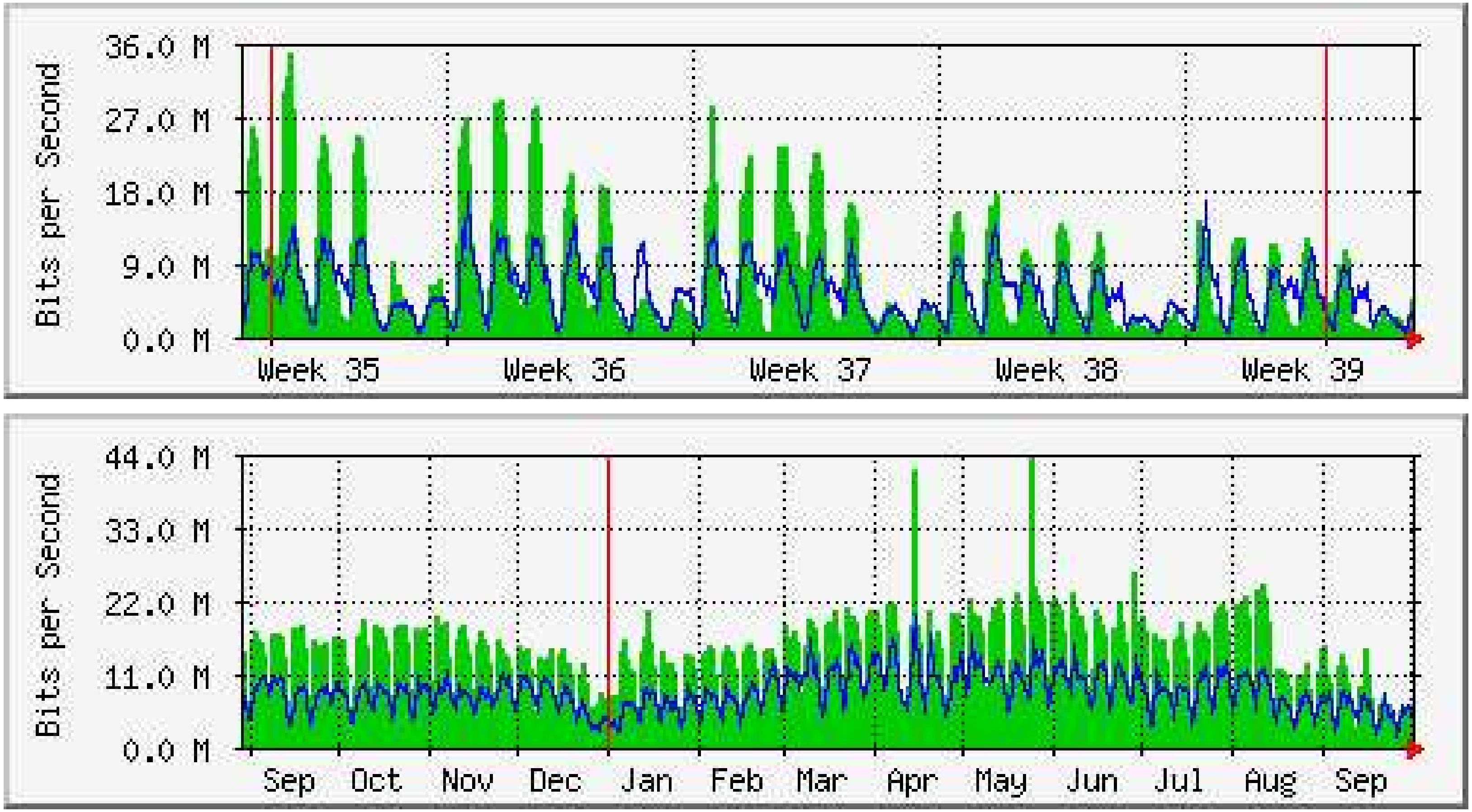




http://www.aarnet.edu.au/network/mrtg.html

Example SNMP data

RRD Tool (or MRTG) are the commonest form of available SNMP data



http://www.aarnet.edu.au/network/mrtg.html

Modeling

Modeling should be

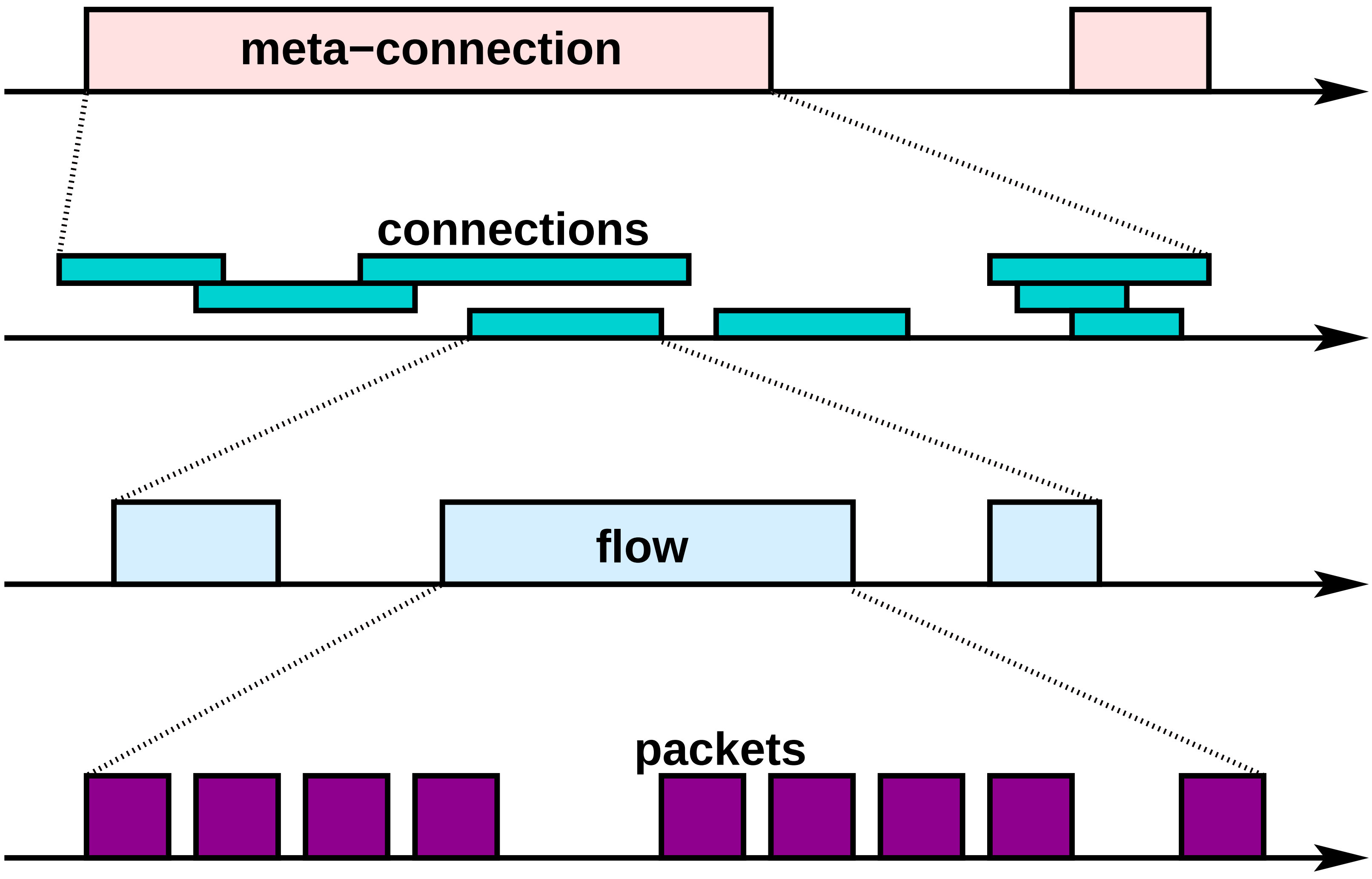
- motivated by the data you can collect
  - SNMP
  - netflow
  - packet traces
  - sampling
- motivated by the application for which it is needed
  - traffic engineering (optimizing routing for existing network and traffic)
  - capacity planning (designing network)
  - protocol design (e.g. TCP congestion control)
- as simple as possible (Occam's razor)

Modeling Goals

- detecting anomalies
  - timescale: minutes
- reliability analysis
  - timescale: hours to days
- traffic engineering
  - timescale: days to weeks
- capacity planning
  - timescale: months
- buffer sizing (in router design)
  - timescale: years

Big Structural Model

model packets, flows, connections, and meta-connections



Big Structural Model

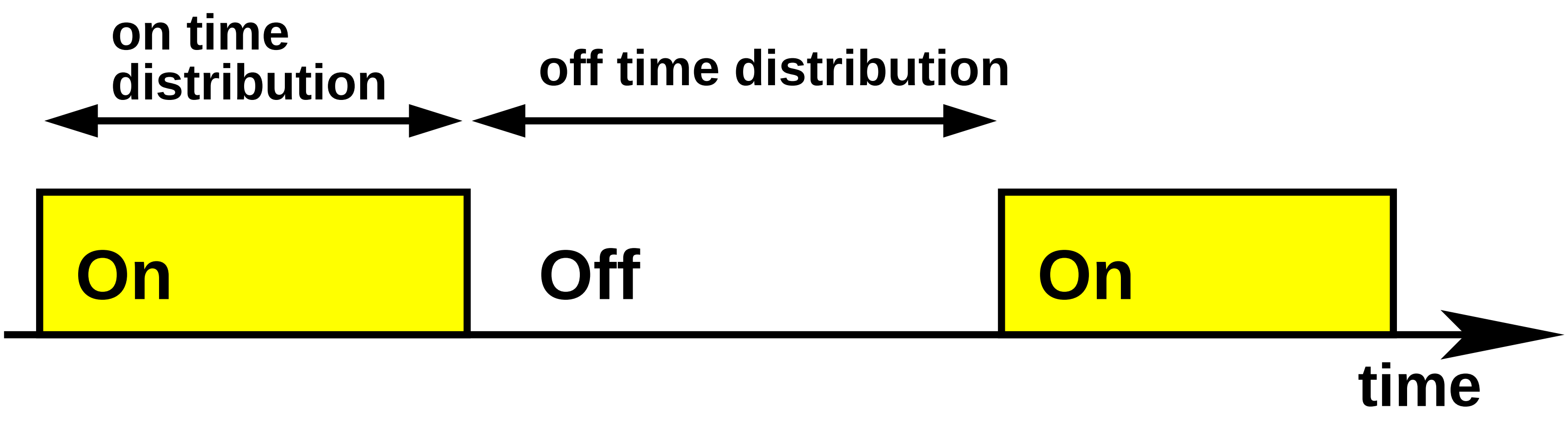
- motivated by explanation of traffic
  - Download a web page
    - HTML page has multiple embedded objects (images etc)
    - under HTTP 1.0, one connection per object
    - packet arrival process within connections governed by TCP congestion control
  - streaming video traffic
    - control connection may be different from data connection
    - two connections (per streaming download)
    - one flow per connection per direction
    - flow $=$ regular stream of video frames

Big Structural Model

- Lots of parameters to estimate
  - meta-connections
    - arrival process
  - connections
    - connections per meta-connection
    - duration and arrival process
  - flows
    - flow structure inside connections
  - packets
    - packet-arrival process inside flows
- Not a parsimoneous model
  - might be OK if we get par.s from the 'physics'
- Makes a lot of assumptions
  - packet arrivals in different connections aren't correlated

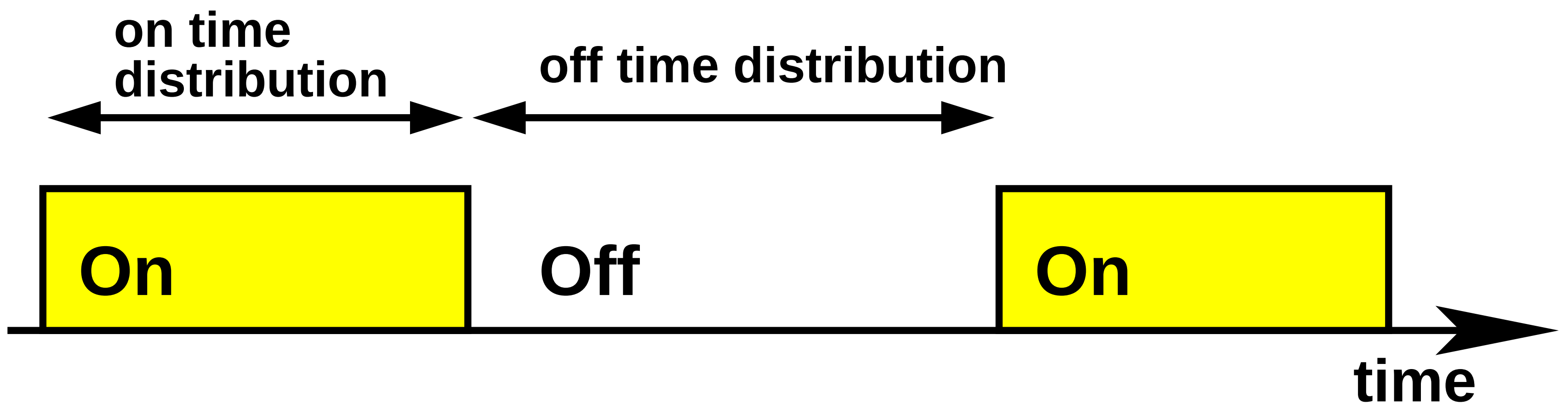
Simple Fluid models

- described by the **rate** $r(t)$ at time t
- the rate can be a random process
- fluid models don't describe individual packet arrivals
- example: On/Off process
  - rate when On is r
  - rate when Off is 0



Simple On/Off process

The Simple On/Off process is a fluid model



- rate when On is r
- rate when Off is 0
- On times are IID random variables
- Off times are IID random variables
- On and Off times are independent
- Forms an **alternating renewal process**

Renewal processes

Take a series of non-negative IID random variables X_k and define a sequence by $T_0 = 0$ and

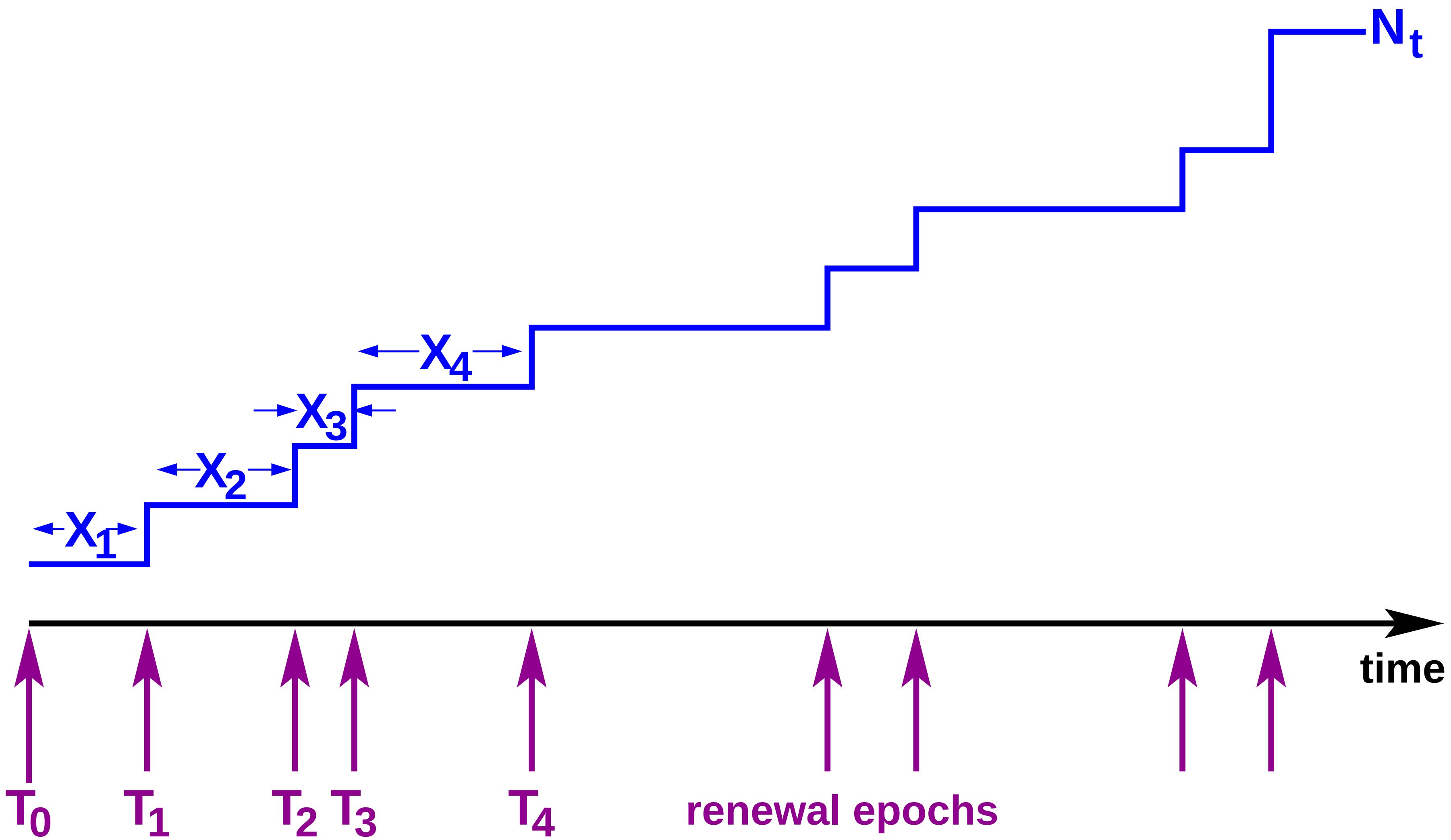
$$T_n = \sum_{k=1}^{n} X_k \quad \text{or} \quad T_n = T_{n-1} + X_n$$

- We call the epochs T_n **renewals**.
- The process “starts again” at each T_n, e.g. we could define a new process T'_n by $T'_0 = T_n$, and

$$T'_m = T_n + \sum_{i=n+1}^{n+m} X_k$$

  with the same statistical behaviour as T_n
- the number of renewals in $[0,t]$ is $N_t = \inf\{k | T_k > t\}$,

Renewal processes



Renewal processes

Assume that X_i has CDF $F(x)$ and mean $E[X_i] = \mu$, and is non-arithmetic (doesn't fall on set values $\delta, 2\delta, \ldots$)

Blackwell's Renewal Theorem: As $t \to \infty$

$$\frac{N(t,t+h)}{t} \to \frac{h}{\mu}$$

where $N(t,t+h)$ is the number of renewal events in the interval $[t,t+h]$

Proof: see "Probability: Theory and Examples", Rick Durret, 3rd Ed, pp. 203-205, Brookes/Cole, 2005.

Renewal processes

Assume that X_i has CDF $F(x)$ and mean $E[X_i] = \mu$, and is non-arithmetic (doesn't fall on set values $\delta, 2\delta, \ldots$), then T_{N_t} is the time of the next renewal after time t.

Residual lifetime: $T_{N_t} - t$ is the waiting time until the next renewal, and as $t \to \infty$ the CDF

$$G(x) = P\{T_{N_t} - t \leq x\} \to \frac{1}{\mu} \int_0^x 1 - F(y)\, dy$$

Proof: see "Probability: Theory and Examples", Rick Durret, 3rd Ed, pp. 205-215, Brookes/Cole, 2005.

Example Renewal processes

The classic example of a renewal process is the Poisson Process, where the X_i are exponentially distributed, e.g. $F(x) = 1 - e^{-\lambda x}$

$$G(x) = \frac{1}{\mu} \int_0^x 1 - F(y)\,dy = 1 - e^{-\lambda x}$$

Bus paradox: a prospective passenger arriving at a bus stop, where the Buses arrive in a Poisson Process has to wait a random time with CDF $G(x)$. But if he just missed a bus, he would have to wait the same amount of time!

Alternating renewal processes

Take two series of non-negative IID random variables X_i and Y_i (which are also independent of each other) with CDFs $F_X(x)$ and $F_Y(x)$. Take $T_0 = 0$

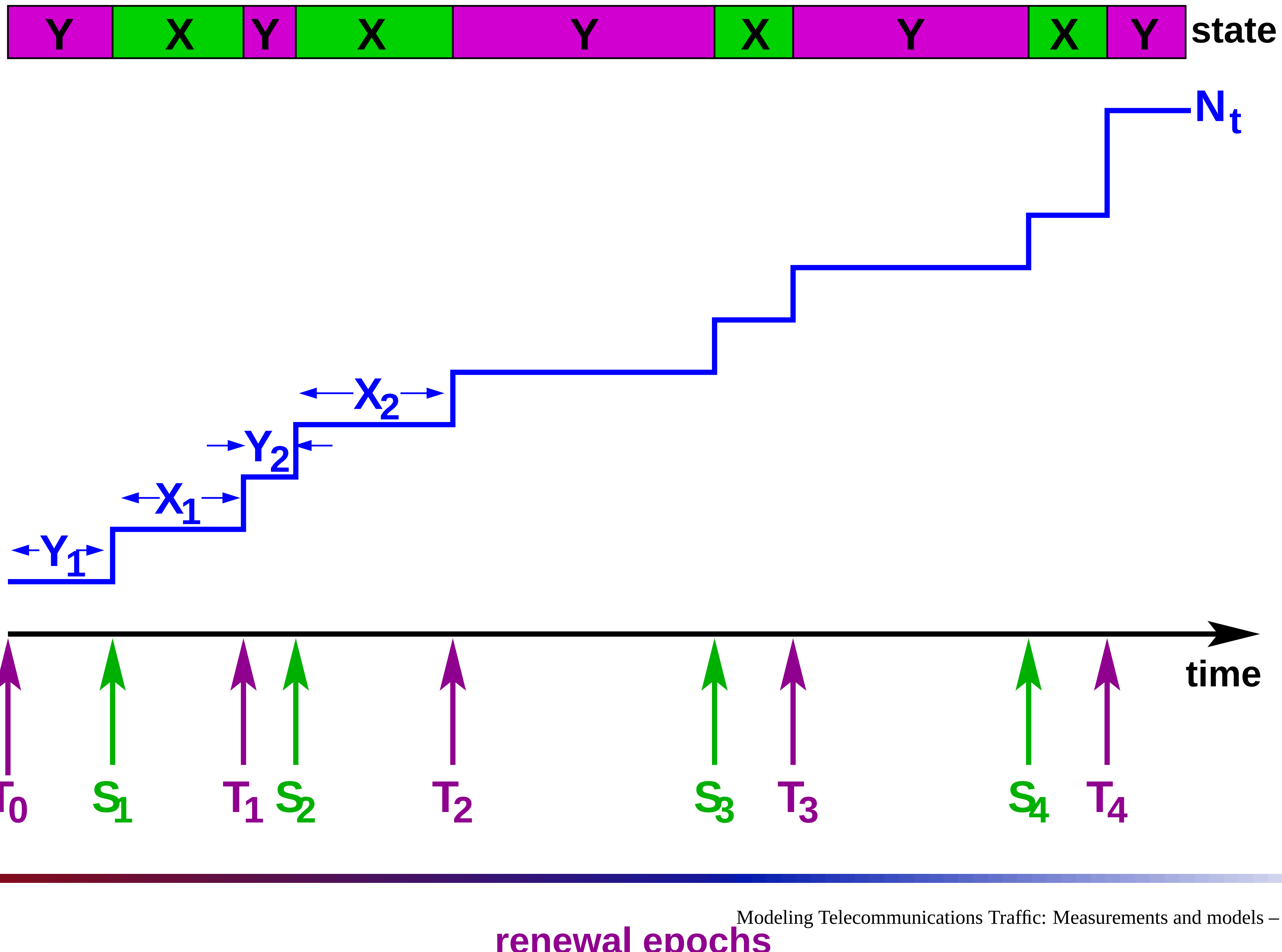
$$\begin{aligned} S_n &= T_{n-1} + Y_n \\ T_n &= S_n + X_n \end{aligned}$$

We say the process is in state X or Y if the last renewal point was at S_n and T_n, respectively.

Given $E[X_i] = \mu_X$ and $E[Y_i] = \mu_Y$, then as $t \to \infty$

$$P\{\text{state} = X\} \to \frac{\mu_X}{\mu_X + \mu_Y}$$

Alternating renewal processes



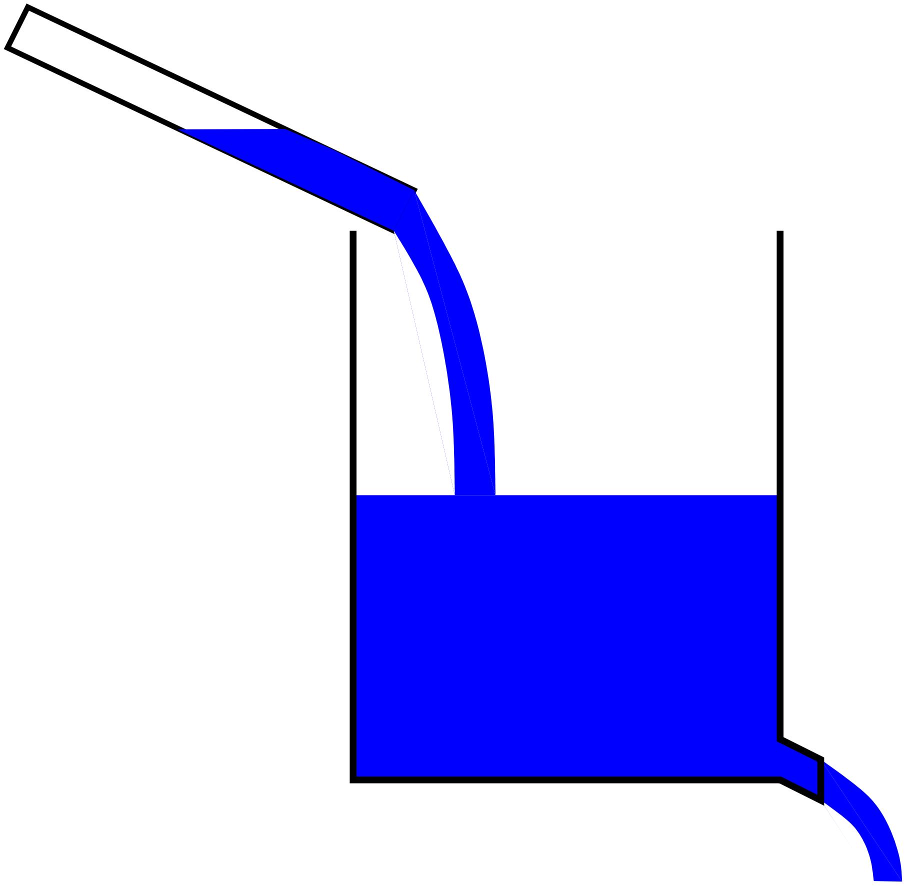
Simple On/Off process

The Simple On/Off process is a fluid model, based on an alternating renewal process, where the On and Off states produce different rates of traffic.

- advantages:
  - few parameters to fit
    - On and Off time distributions
  - matches some sets of data reasonably
- disadvantages:
  - only two rates
  - limited correlations in process
  - requires packet level data
    - On and Off times don't correspond to flows
  - doesn't generate packet arrivals

Simulations using fluid models

- fluid simulation
  - e.g. leaky bucket



- generate a point process from the rates
  - doubly stochastic point process

Doubly stochastic point process

Doubly Stochastic Poisson Processes are inhomogeneous Poisson Processes, where the rate is controlled by a stochastic process. E.G. use a fluid model to give rate of a Poisson process.

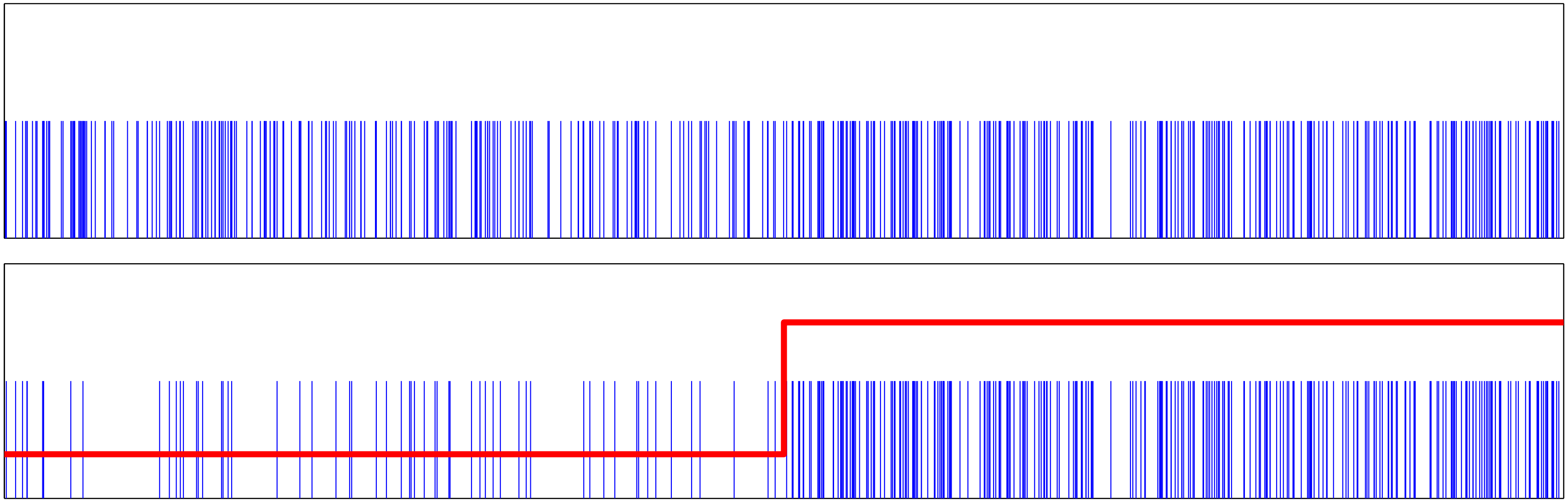
Examples:

- Interupted Poisson Process
  Rates drawn from an On/Off process
- Markov Modulated Poisson Process
  Rates depend on an On/Off process
- Shot Noise Poisson Process
  Rates from a Poisson Process passed through EWMA

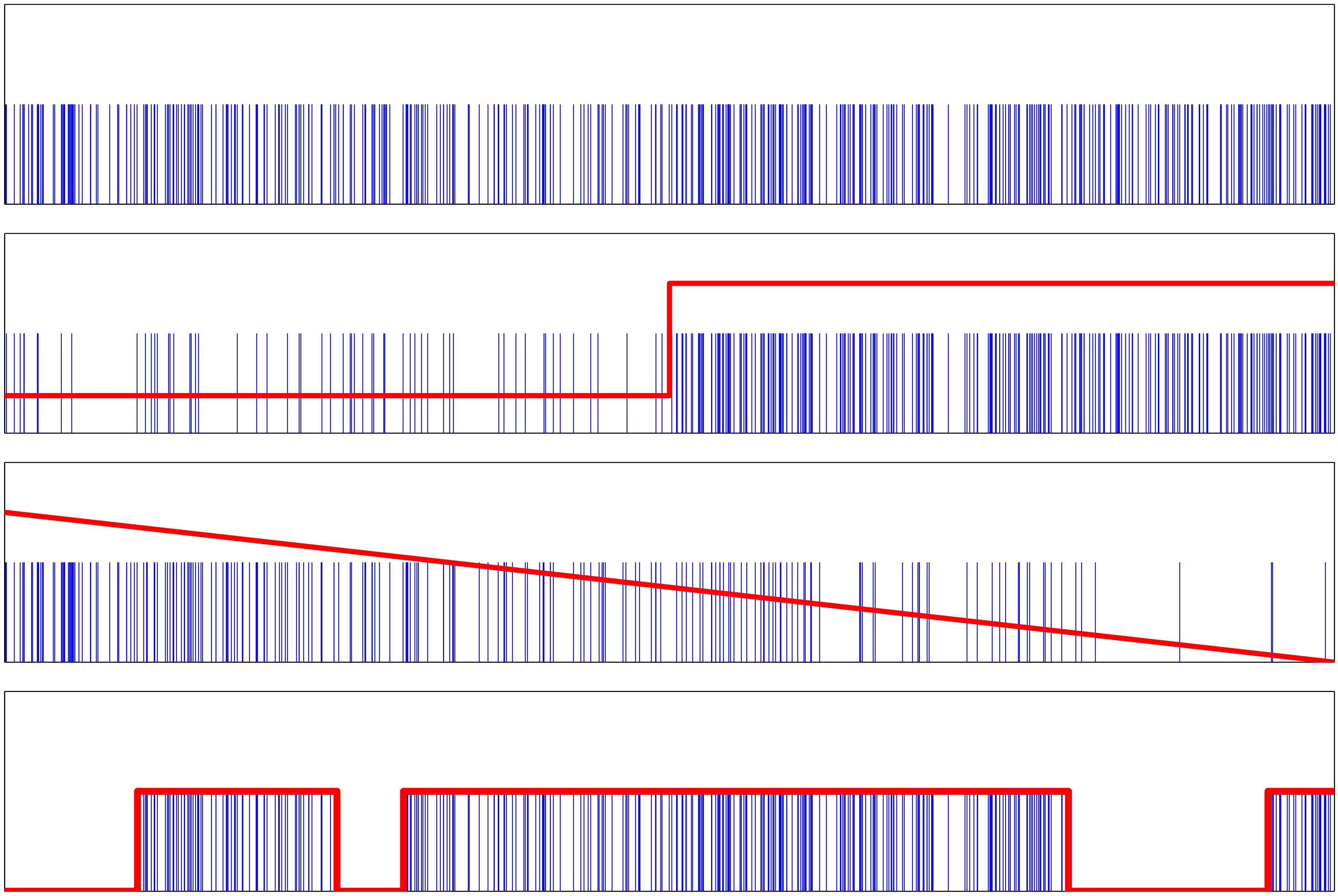
Inhomogeneous PP Generation

One method for generation of Inhomogeneous Poisson Processes is

- generate a homogenous PP with $\lambda = \sup_t \lambda(t)$
- The PP is a series of points at times t_i
- discard each point with probability $1 - \lambda(t_i)/\lambda$



Inhomogeneous PP Generation



Renewal Reward Process

Take a renewal process, and associate a random **reward** with each renewal time. Take the fluid rate at time t to be the current reward.

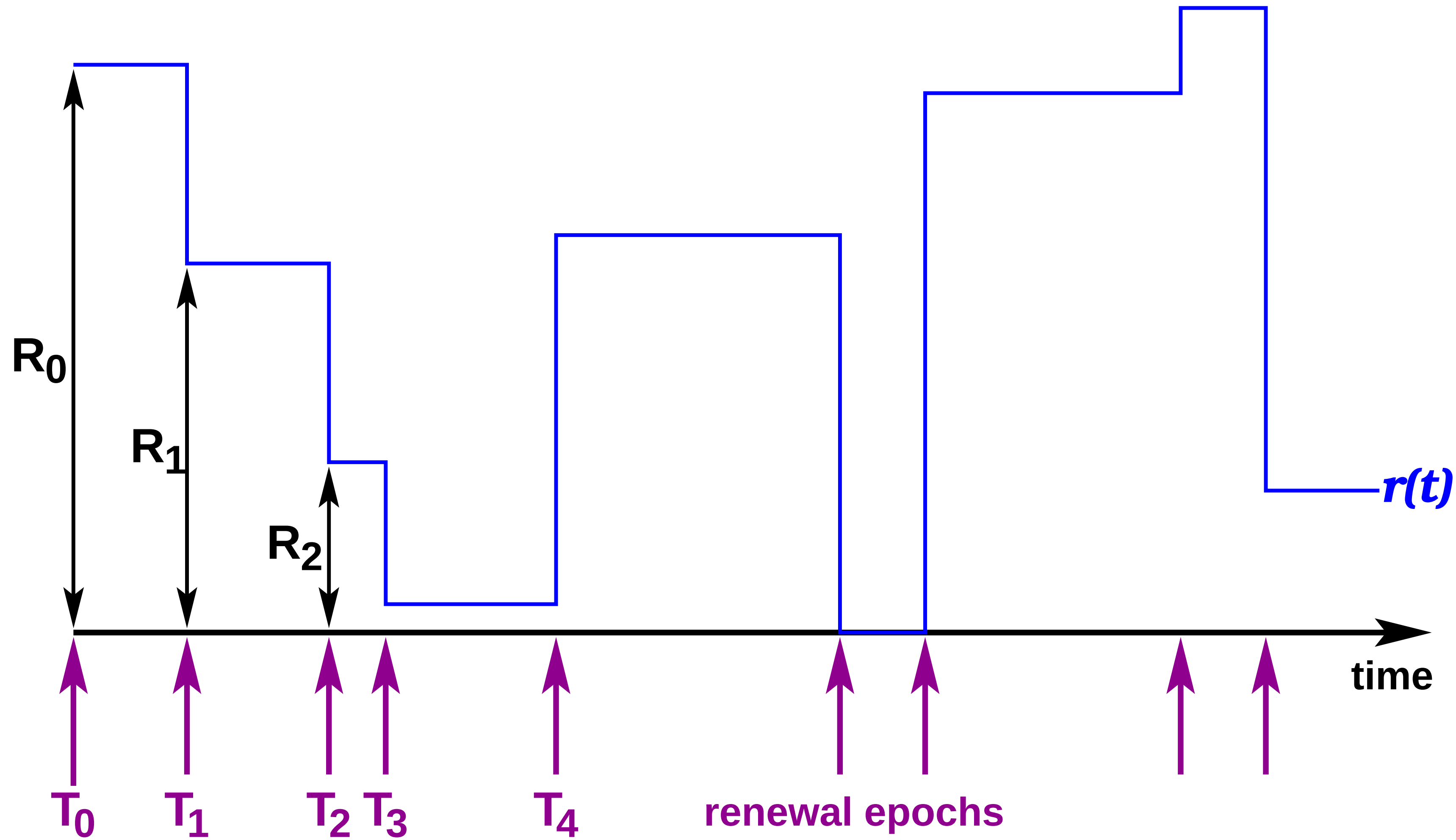
More precisely:

- Take a renewal process generated by a series of non-negative IID random variables X_k, i.e. $T_0 = 0$ and

$$T_n = \sum_{k=1}^{n} X_k \quad \text{or} \quad T_n = T_{n-1} + X_n$$

- Take a series of non-negative IID random variables R_k to be the reward at renewal epoch k.
- Take the traffic rate at time t to be $r(t) = R_{N_t - 1}$

Renewal Reward Process



Renewal Reward Process

- advantages:
  - few parameters to fit
    - renewal time distributions
    - reward distributions
  - matches some sets of data reasonably
    - more possible rates
- disadvantages:
  - limited correlations in process
  - requires packet level data
    - renewal times don't correspond to flows
  - doesn't generate packet arrivals
  - no correlations between rewards and renewal times

Markov Renewal Process

Generalize renewal process to allow for many states.

- Define a Markov chain on states $\mathcal{S}$, with probability transition matrix $P = (p_{ij})$.
- Define a set of CDFs $F_{ij}(t)$, for all $p_{ij} > 0$. The time spent in the current state i conditioned on the next transition being to state j is given by CDF $F_{ij}(t)$.
- The total state is described by (J_n, X_n), where J_n is the state after n transitions, and X_n is the time spent in state J_{n-1} before the transition to state J_n.
- as before $T_n = \sum_k X_k$
- the future behaviour of system depends on the past, only through the current state.

Superposition of On/Off Processes

Take N independent (alternating) renewal processes, and at time t let the total traffic rate $r(t)$ be given by

$$r(t) = \sum_{i=1}^{N} r_i(t)$$

where $r_i(t)$ is the rate of the ith process at time t.

- moving from single source models to aggregate traffic models
  - a model of N sources
  - simplest case, assume all sources are identical
- aggregate traffic models are often more robust, but don't allow for individual dynamics of applications

Superposition of On/Off Processes

- assume On/Off sources are identical and independent, with rate r when On.
- the number of On sources follows a binomial distribution
  - for $p = p\{On\}$, the distribution of On sources

$$p\{N_{On} = n\} = \binom{N}{n} p^n (1-p)^{N-n}$$

- for large N, and $p\{On\}$ and $p\{Off\}$ not too small, we get an approximately Gaussian process
  - mean traffic rate = rNp, and variance $r^2Np(1-p)$.
  - correlations are governed by the On and Off time distributions

Gaussian processes

- advantages:
  - few parameters to fit
    - mean, variance
    - autocorrelation function
  - matches some sets of data reasonably
    - aggregates
  - no longer necessarily requires packet data
- disadvantages:
  - doesn't generate packet arrivals
  - marginal distribution is constrained to be Gaussian
  - doesn't describe individual applications well

What's important to get right

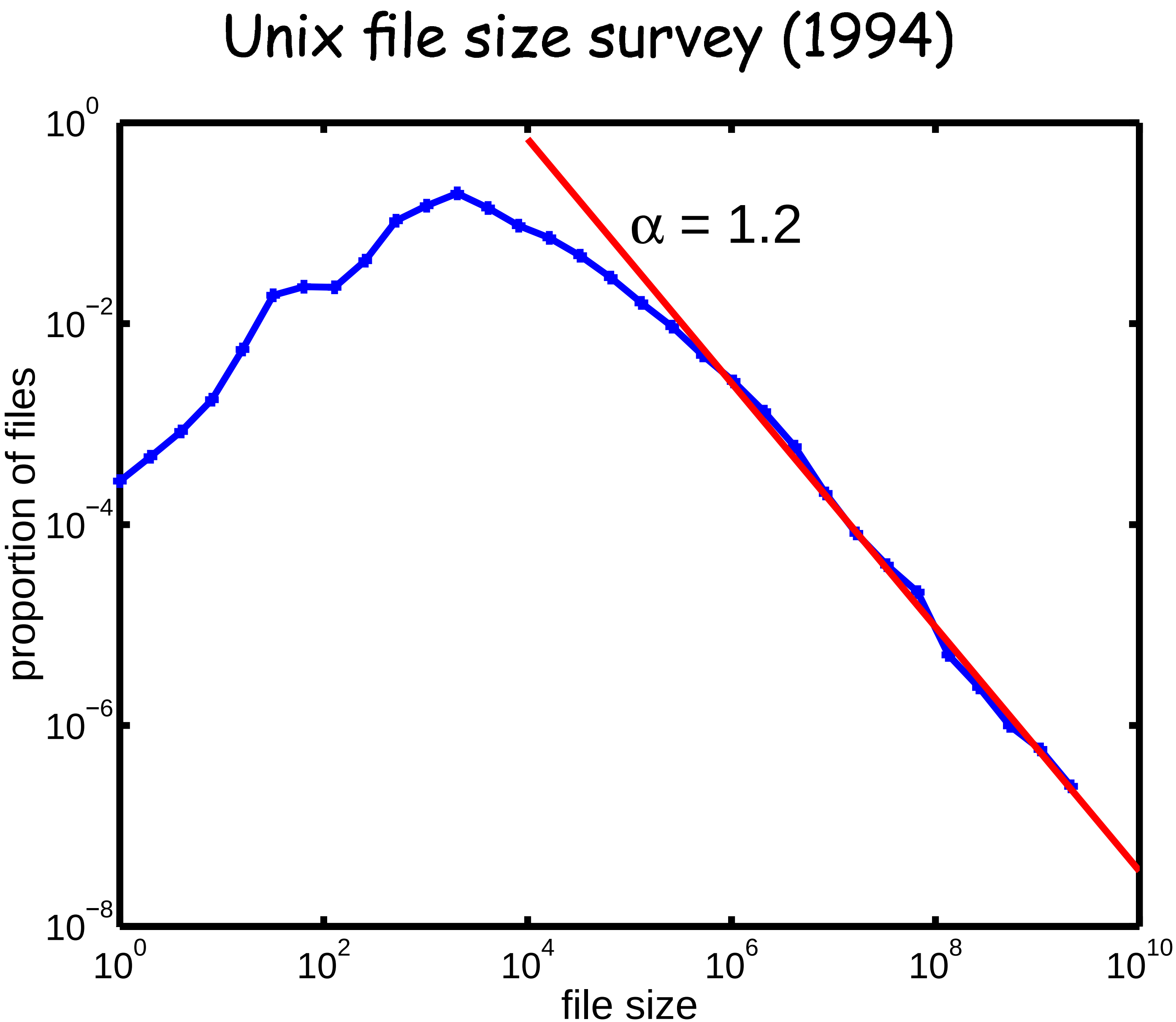
When modeling we need to get some bits right, e.g.

- distributions of renewal times!
- values of renewal rewards

Both exhibit heavy-tails!!!!

- the tail events have tiny probabilities, but still have a profound impact on the overall behaviour
- infinite variance!

Heavy-tails



Unix file size survey (1994)
$\alpha = 1.2$
proportion of files
file size
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^0
10^2
10^4
10^6
10^8
10^{10}

Fat-tails

Important not to confuse heavy-tails with fat-tails.