Communications Systems Performance

Revision notes by Michael Prior-Jones, based on course by Dave Pearce

Useful things to know:

The Central Limit Theorem:

The sum of a very large number of independent random variables has a Gaussian (Normal) distribution.

Adding distributions:

- The mean of the sum of two independent random variables is the sum of the means of the variables.
- The variance of the sum of two independent random variables is the sum of the variances of the variables.
- The distribution of the sum of two independent random variables is the convolution of the distribution of the variables.
- The convolution of two normal distributions is another normal distribution.

Exponential & Poisson Distributions

If you have:

- a large number of independent events
- each event has the same property of occurring
- the event is of negligible length

then the *number of events occurring in a given time* is a Poisson distribution. The *time between events* is an exponential distribution.

Exponential distribution:

$$P(T)dt = l\ \exp(-lT)$$

Mean of the exponential distribution:

$\bar{x} = \frac{1}{l}$ i.e. mean time between events.

Poisson distribution:

$$P(m) = \frac{(lT)^m}{m!}\exp(-lT)$$

This is the probability of m events occurring within time interval T, given that the probability of an individual event occurring is λ.

Mean of the Poisson distribution:

$\bar{x} = lT$

and, interestingly, the variance is the same.

Service Requirements

Traffic types:

- Synchronous: generates at fixed rate according to an external clock
- Asynchronous: generates data irregularly
- Isochronous: generates data at an almost fixed rate according to an internal clock.

Bit-rates: maximum, minimum, constant
Delay (latency): average time between Tx and Rx.
Jitter: standard deviation of delay.
Error rate: bit error rate or packet error rate.

All factors depend on traffic: voice telephony, video telephony, voice broadcast, video broadcast, continuous data, file transfer, interactive gaming, etc, etc.

Optimum packet size is dependant on BER and packet header size. Forward Error Correction codes can be used to improve performance.

Traffic Models

Poisson distributions are traditionally used to model traffic, particularly in the telephone network. However, traffic on many networks, particularly the Internet, is *self-similar*, and doesn't average out to a constant rate over a long period of time.

Self similarity is measured by *Hurst's parameter*. To do this, count the events in a given period T many times. Take the range and divide by the standard deviation. This gives you the *re-scaled range*, R. Plot against T and fit it to the following relationship.

$$R = kT^{a}$$

Alpha is Hurst's parameter, which varies between 0.5 and 1. A value of 0.5 indicates that the traffic is not self-similar.

A common model for self-similar traffic is the Pareto distribution. When you use this to determine the intervals between periods of packet transmission and periods of silence you get a reasonably accurate self-similar traffic model.

Empirical traffic models are the last resort: just choose a standard data set to send. This is common in video compression research: there are standard international test slides for stills, and an increasing tendency for US researchers to use Star Wars as a benchmark …

Packet Switching and Queuing

The M/M/1 queue:

- exponential interarrival times
- exponential servicing time
- single queue

Little's formula

Calculates the average number of packets in the queue:

$$w_{mean} = lT_w$$

where λ is the rate of arrivals and T_w is the average time a packet spends waiting.

How to solve Markov chain problems:

- Draw the chain and label it with all the probabilities
- Use the assumption that average number of transitions between adjacent states must be equal in both directions (i.e. queue is in equilibrium).

Also apply constraint that sum of the probabilities of being in each state must come to 1, as system must be in one of the states.

The M/M/1/K queue:

As above, but has a finite length K.

Discard strategies:

When the queue is full you must discard packets. Intelligent discard strategies involve dropping packets from "real time" streams, as there's no time to retransmit. On the other hand, data packets should not be dropped as they generate additional traffic in retransmissions.

Circuit Switching: Erlang and Engset distributions

Erlang-B distribution:

Models the telephone system

- very large number of users
- users use phone infrequently
- if the users are blocked, they don't retry

Erlang-C distribution:

more realistic model of public phone system

- as Erlang-B, but blocked calls are queued until a line is available.

Engset distribution

appropriate for small circuit-switched networks

- as Erlang-B, but number of users is not large.

Multimedia Traffic on the Internet

Differentiated Services in IP

Priority field in header: routers know that certain packets can be dropped.

RSVP: resource reservation protocol

Connection-orientated: allocate space in routers before sending traffic. Can waste space. Only works if all routers support it. Uses extra processing power. Guarantees Quality of Service.

RTP: real time protocol

Used for streaming audio/video. Uses UDP datagrams and timestamps.

Congestion control in TCP

Error control policies:

Acknowledge policies:

- immediate: acknowledge immediately
- cumulative: wait, see if more packets arrive, acknowledge all packets in one go.

Retransmit policies:

- First only: retransmit first unacknowledged packet only
- Batch: send all unacknowledged packets again
- Individual: maintain separate timers for each packet, and retransmit when they time out.

Accept policies:

- In order only: if you get a packet out-of-order, chuck it away.
- In window: maintain Rx buffer and wait for missing packets to arrive.

Congestion avoidance in TCP

Slow Start:

Start TCP with window size of 1. Double window size every time you get an acknowledgement until you reach the negotiated size.

Dynamic Window sizing:

When a packet is lost, go back to a window size of 1. Go through slow start until you get to 50% of the window size at which the packet was lost. Then increase linearly for each packet acknowledged.