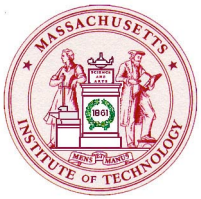
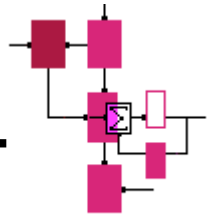




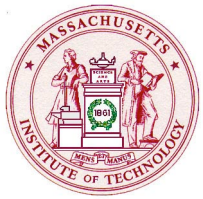
Delay Models and Queueing

Muriel Medard

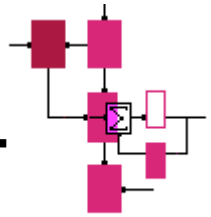
EECS

LIDS

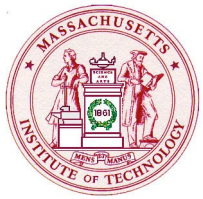
MIT



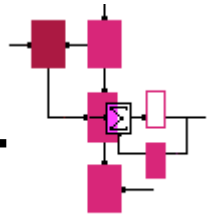
Outline



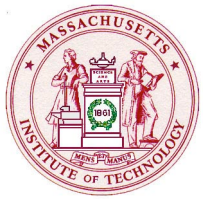
- Introduction
- Concepts of ergodicity
- Little's theorem
- Application: Pollaczek-Khinchin
- Poisson process
- Markov overview
- M/M/… systems
- Review of transforms
- M/G/… systems
- Extensions of Markov approach to non M/M/ systems
- Reverse chains
- Burke's theorem for reversibility
- Networks of queues
- Kleinrock's independence assumption
- Jackson's Theorem



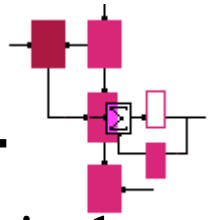
The importance of queues



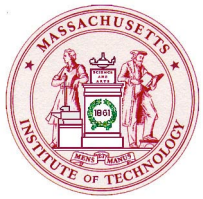
- When do queues appear?
  - Systems in which some serving entities provide some service in a shared fashion to some other entities requiring service
- Examples
  - customers at an ATM, a fast food restaurant
  - Routers: packets are held in buffers for routing
  - Requests for service from a server or several servers
  - Call requests in a circuit-oriented system such as traditional telephony, mobile networks or high-speed optical connections



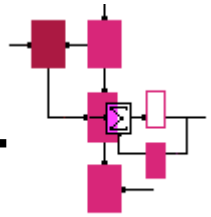
What questions may we want to pose?



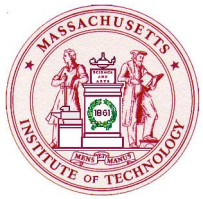
- What is the average number of users in the system? What is the average delay?
- What is the probability a request will find a busy server?
- What is the delay for serving my request? Should I upgrade to a more powerful server or buy more servers?
- What is the probability that a packet is dropped because of buffer overflow? How big do I need to make my buffer to maintain the probability of dropping a packet below some threshold? What is the probability that I cannot accommodate a call request (blocking probability)?
- For networked servers, how does the number of requests queued at each server behave?
- We shall keep these types of questions in mind as we go forward



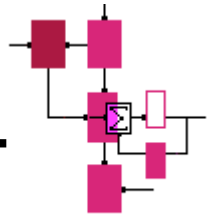
Analysis versus simulation



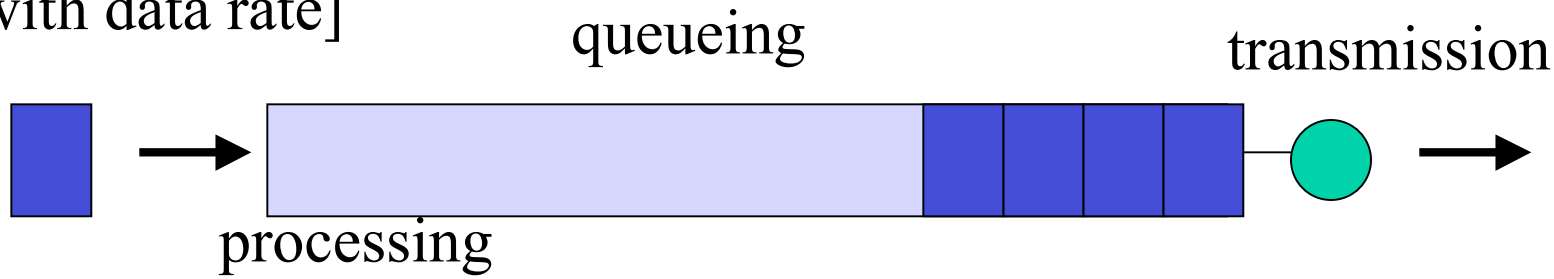
- Why can't I just simulate it?
- Analysis and simulation are complementary, not opposed
- It is generally impossible to simulate a whole system- we need to be able to determine the main components of the system and understand the basis for their interaction
- What are the important parameters? What is their effect?
- In many systems simulation is required to qualify the results from analysis, to obtain results that are too complex computationally

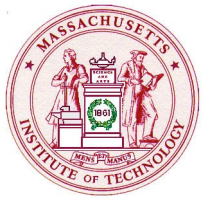


Delay components

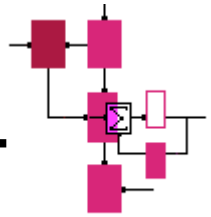


- Processing delay: for instance time from packet reception to assignment to a queue (generally constant)
- Queueing delay: time in queue up to time of transmission
- Transmission delay: actual transmission time (for instance proportional to packet length)
- Propagation delay: time required for the last bit to go from transmitter to receiver (generally proportional to the physical link distance, large for satellite link) [Not to confuse with latency, which is number of bits in flight, latency goes up with data rate]

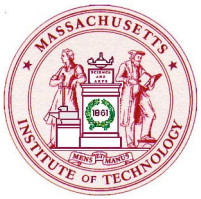




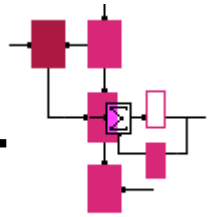
Little's theorem



- Rather than refer to packets, calls, requests, etc… we refer to customers
- Relates delay, average number of customers in queue and arrival rate (λ)
- Little's Theorem: average number of customers = λ x average delay
- Holds under very general assumptions



Main parameters of a queueing system



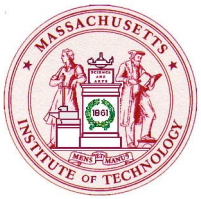
- N(t): number of customers in the system at time t
- P(N(t) = n) = probability there are n customers in the system at time t
- Steady state probability:

$$P_n = \lim_{t \to \infty} P(N(t) = n)$$

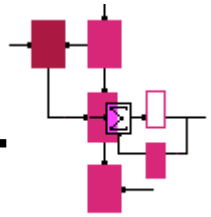
- Mean number in system at time t:

- Time average number in the system:

- We assume the system is **ERGODIC**:

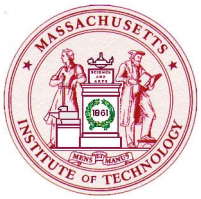


Main parameters

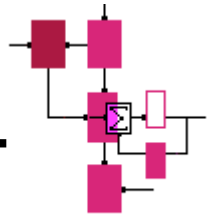


- We looked at the system from the point of view of the customers in it, let us now consider the delay of those customers
- T(k): delay of customer k
- $\alpha(t)$: number of customer arrivals up to time t
- $\beta(t)$: number of customer arrivals up to time t
- Our ergodicity assumption implies that the long-term arrival rate is the long-term departure rate:

- Our ergodicity assumption implies that there exists a limit:

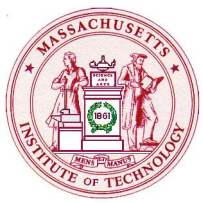


Little's theorem

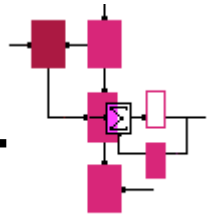


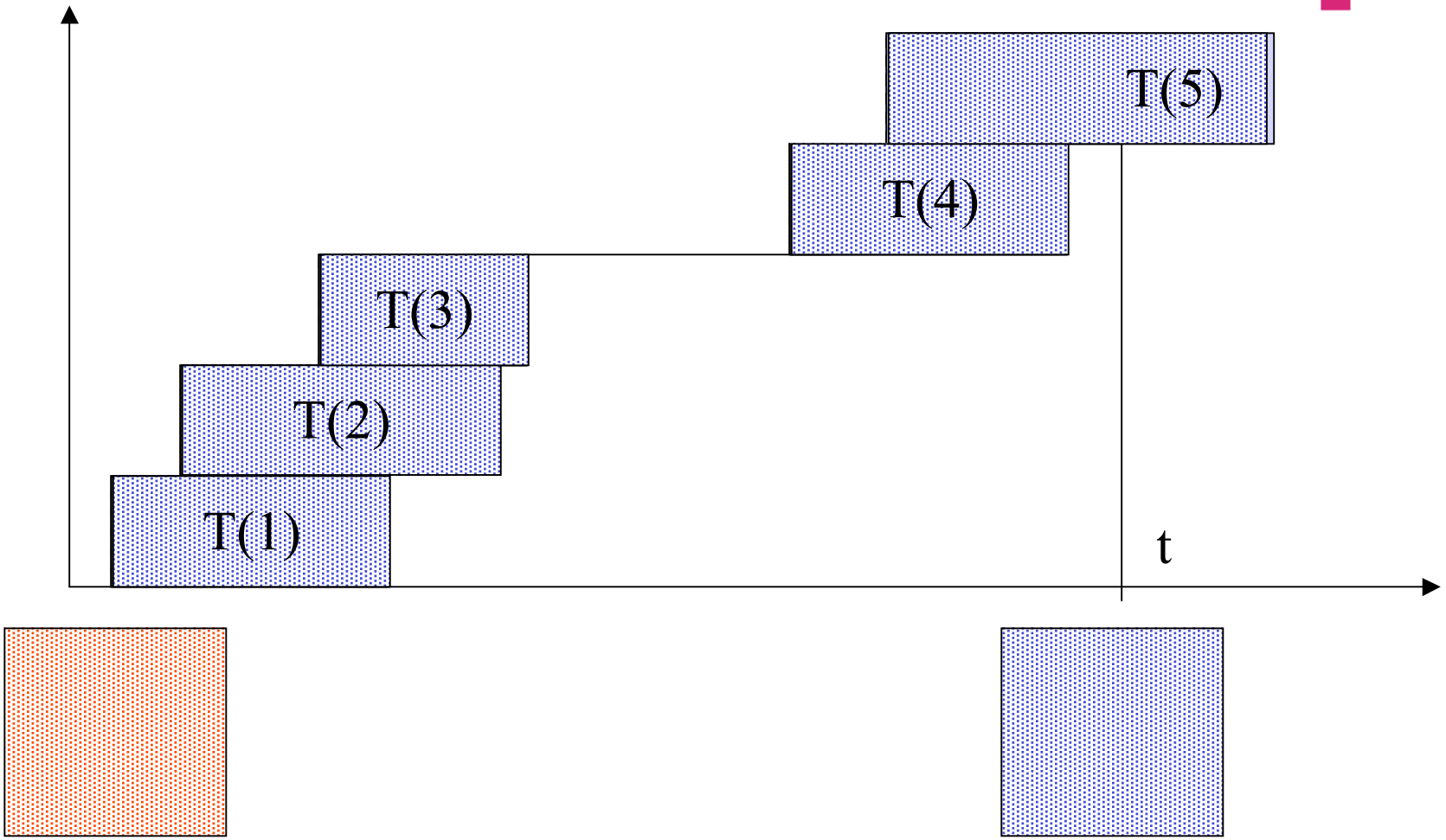
- We have:

- Little's theorem applies to any arrival-departure system with appropriate interpretation of average number of customers in the system, average arrival rate and average customer time in system

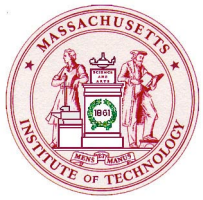


Justification of Little's theorem

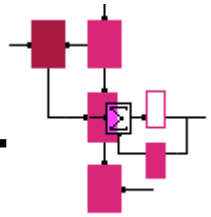




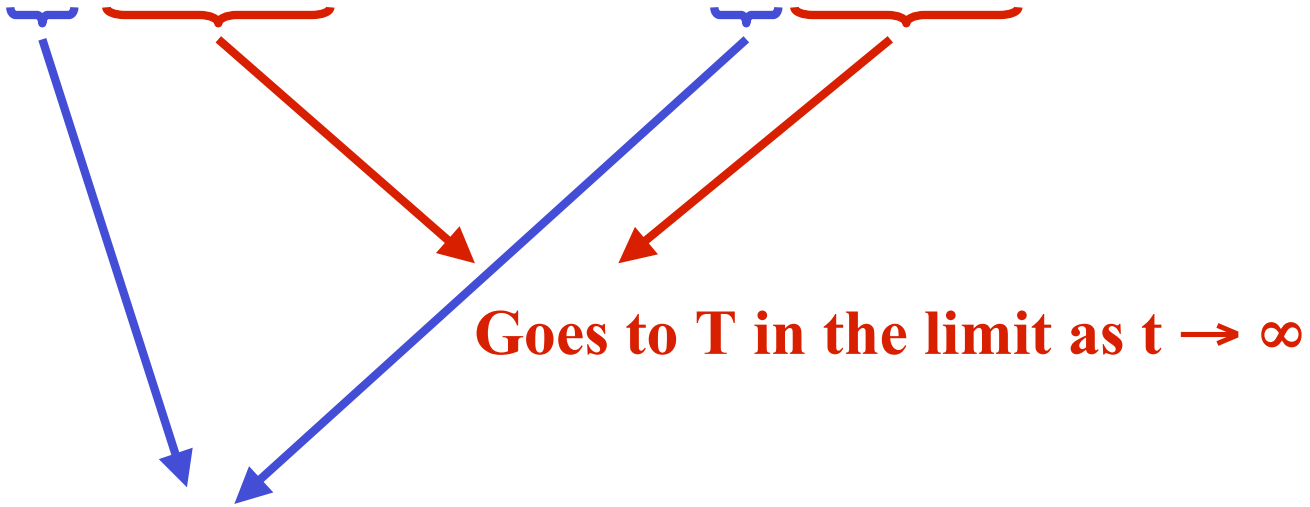
Note: a similar picture holds even if we do not have FIFO

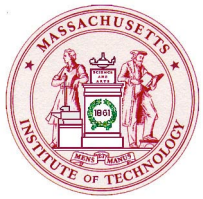


Justification of Little's theorem

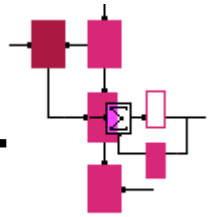


- Taking the average over time:



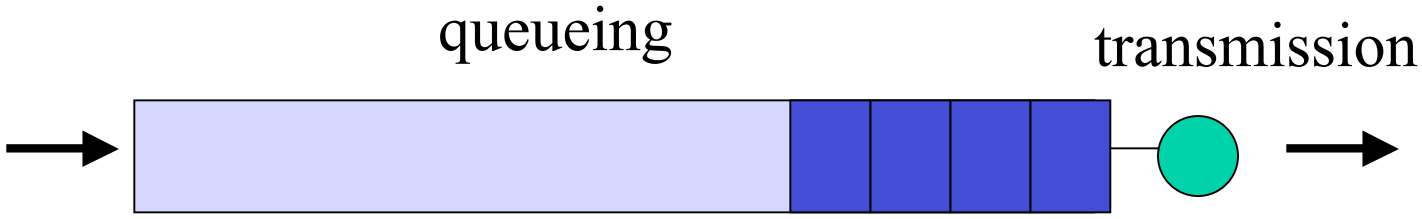


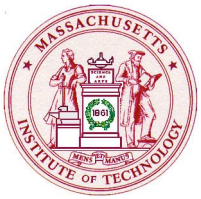
Example of application



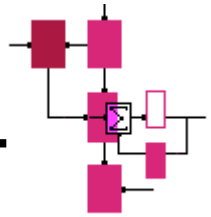
- Applying Little's theorem to the queueing portion to obtain the average number waiting in queue

- Applying Little's theorem to the transmission portion to determine the proportion of time transmission is occurring:



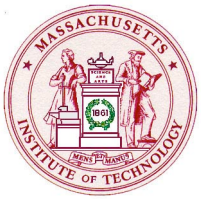


Application to complex systems

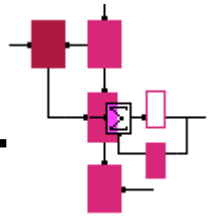


- Suppose we have several different traffic streams
- Applying to each traffic stream yields
- Applying to all M streams collectively:

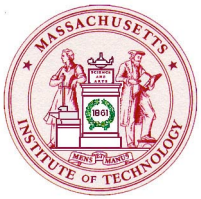
- We have answered the first question insofar as we can determine N from T or T from N, but we need more information to determine both



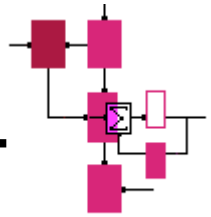
Pollaczek-Khinchin



- General type of service times, X is the service time, with means E[X] and $E[X^2]$ known
- Arrivals are memoryless: arriving customer in steady-state sees mean behavior
- The main formula is the Pollaczek-Khinchin formula (P-K) for the waiting time $\overline{W}$ in queue



Examples



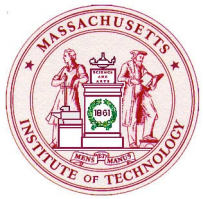
- Nomenclature for queues:

  Arrival type/Service type/Number of servers/Buffer size

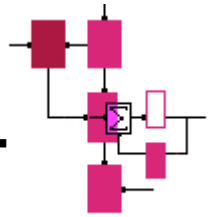
- Common types: M (Poisson distributed), G (general), D (deterministic)
- M/M/1: P-K gives

- M/D/1: deterministic service time where every user has equal service time $1/\mu$, P-K gives

- P-K is a M/G/1 formula

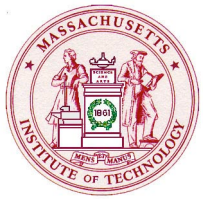


Proof of P-K

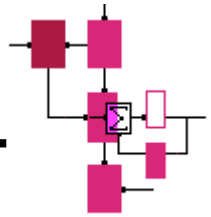


- W(k): waiting time in queue of customer k
- R(k): residual service of customer k
- X(k): service time of customer k
- N(k): number of customer found waiting in queue by customer k (memoryless assumption)

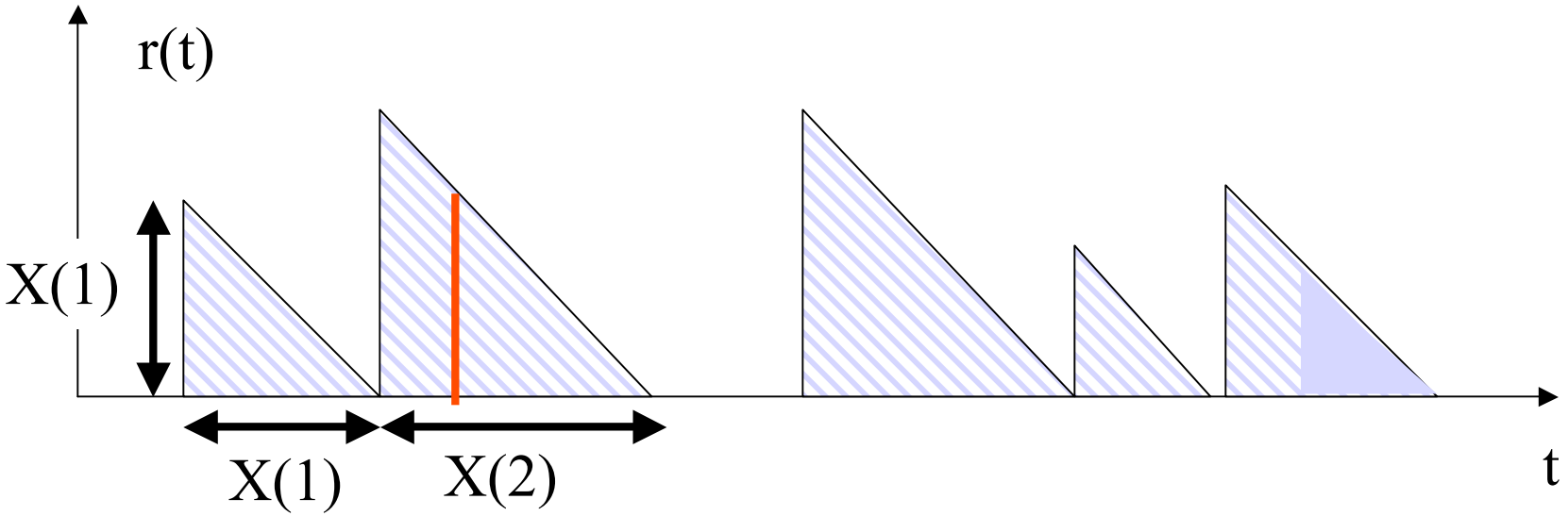
Using Little's theorem

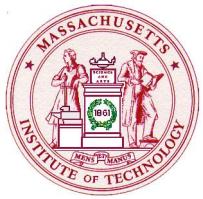


Residual time

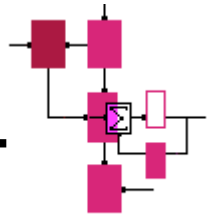


- We have reduced everything in terms of R
- Let r(t) be the residual time at time t,
- Let M(t) be the customer completing service at time t

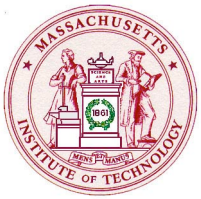




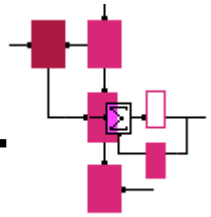
Residual time



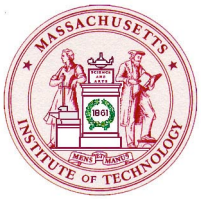
- By looking at the area we consider:



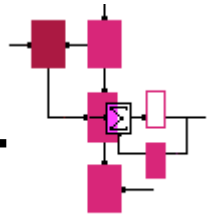
Priorities



- The M/G/1 model will help us understand priorities
- Priorities 1,.., k, with 1 being the highest
- $\lambda(k)$, $\mu(k)$: arrival and service rates of priority k
- W(k): average queuing time for priority k
- $\rho(k) = \lambda(k)/ \mu(k)$: utilization factor for priority k
- $\overline{R}$: mean residual time
- Let us look at preemptive priority:
  - customer in service is interrupted by an arriving higher priority customer
  - Interrupted customer resumes service from the point of interruption

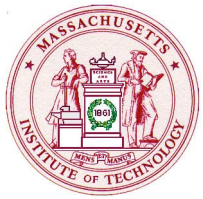


Preemptive priority

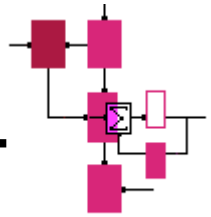


- For preemptive priority:

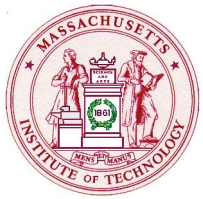
- Higher priority users do not “see” lower priority users



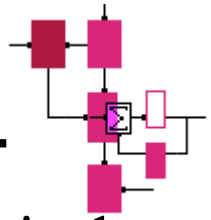
Non-preemptive priority



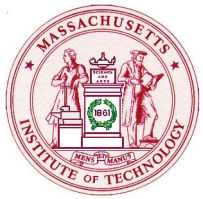
- Customer in service is not interrupted, so W for low priority will depend on high priority
- Assuming <1

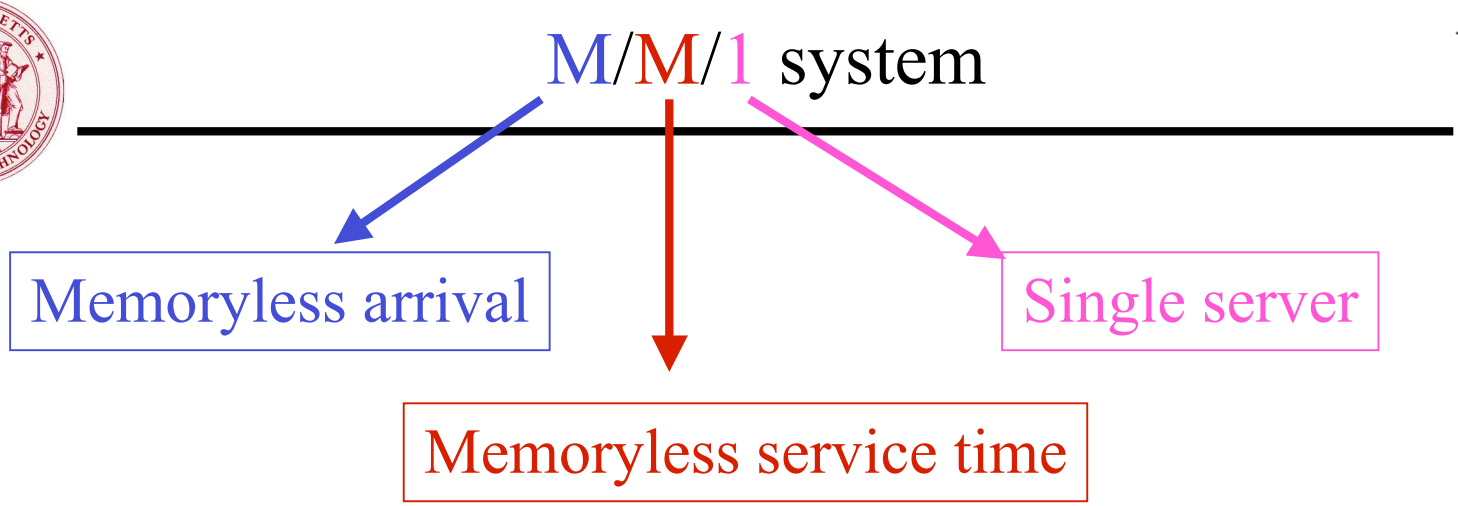


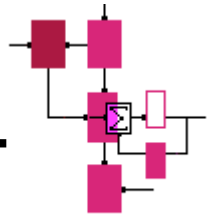
What questions may we want to pose?



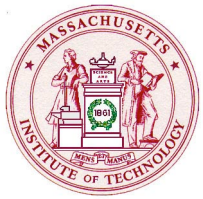
- What is the average number of users in the system? What is the average delay?
- What is the probability a request will find a busy server?
- What is the delay for serving my request? Should I upgrade to a more powerful server or buy more servers?
- What is the probability that a packet is dropped because of buffer overflow? How big do I need to make my buffer to maintain the probability of dropping a packet below some threshold? What is the probability that I cannot accommodate a call request (blocking probability)?
- For networked servers, how does the number of requests queued at each server behave?
- We shall keep these types of questions in mind as we go forward



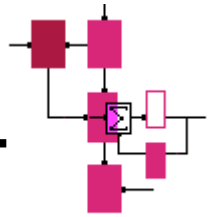




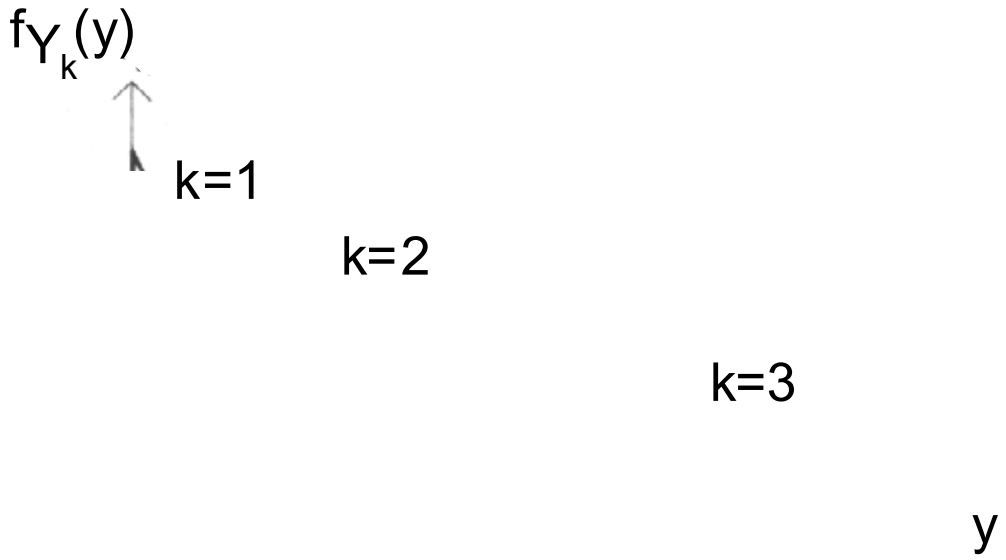
- Poisson process A(t) with rate λ is a probabilistic arrival process such that:
  - number of arrivals in disjoint intervals are independent
  - number of arrivals in any interval of length τ has Poisson distribution with parameter $\lambda\tau$:

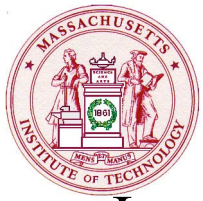


Poisson process

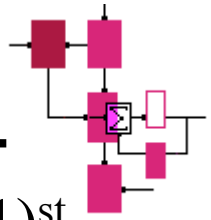


- The distribution of the time Y_k of the k th arrival has an Erlang probability density function:





Further properties of the Poisson process



- Interarrival time $\tau(n)$ for interarrival between n^{th} and $(n+1)^{st}$ arrival. The interarrival times are independent and exponentially distributed with parameter λ:

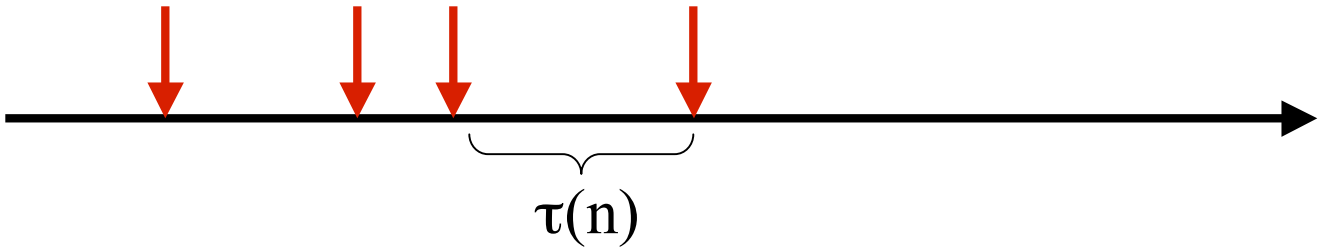
- For very small δ, we can say that:

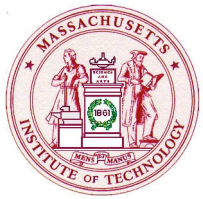
$$P(A(t+\delta) - A(t) = 0) = 1 - \lambda\delta + o(\delta)$$

$$P(A(t+\delta) - A(t) = 1) = \lambda\delta + o(\delta)$$

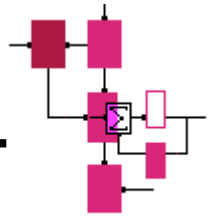
$$P(A(t+\delta) - A(t) \geq 2) = o(\delta)$$

$$\text{where } \frac{o(\delta)}{\delta} \to 0 \text{ as } \delta \to 0$$





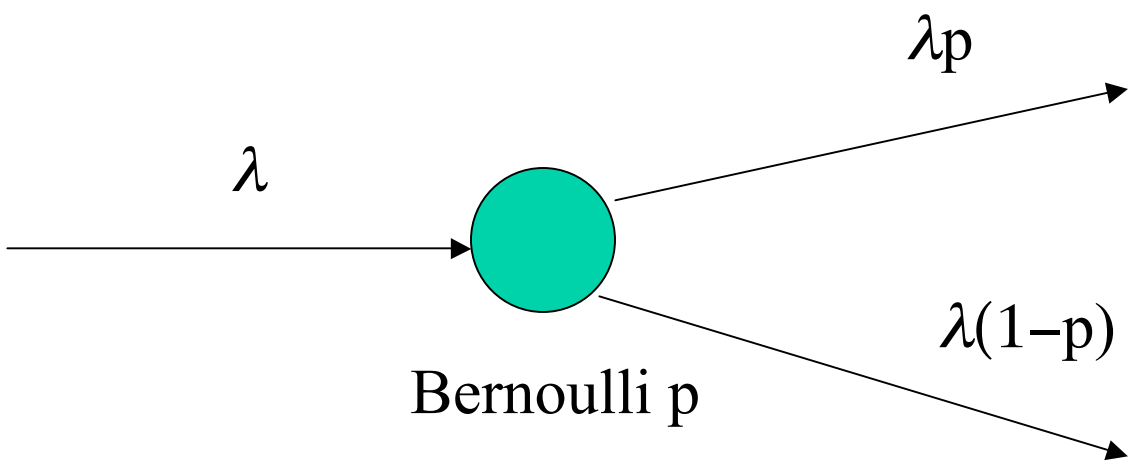
Further properties of the Poisson process



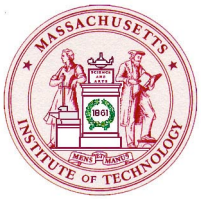
- We can condition on any past history, the time until the next arrival is always exponentially distributed, hence the term memoryless
- If $A_1, A_2, \ldots, A_k$ are independent Poisson processes with rates $\lambda_1, \lambda_2, \ldots, \lambda_k$ then the process $A = A_1 + A_2 + \ldots + A_k$ is Poisson with rate $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_k$
- Poisson processes are generally a good model for the collective traffic of a large number of small users

Further properties of the Poisson process

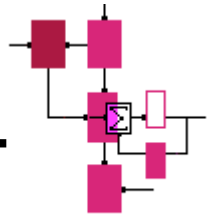
- Probabilistic splitting of Poisson processes is Poisson



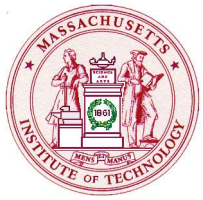
- What is probability that last arrival was from the top process?



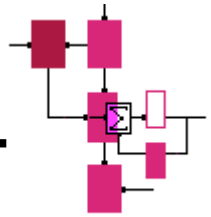
M/M/1



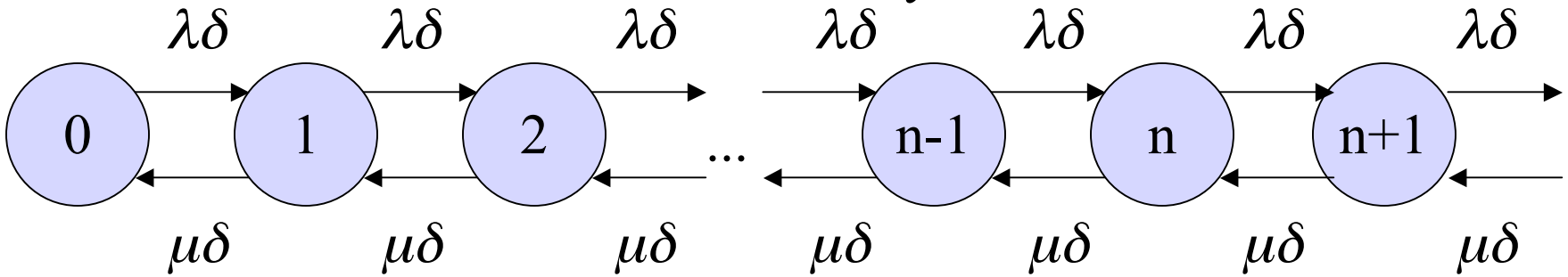
- Single server
- Poisson arrival process with rate λ
- Independent identically distributed (IID) service times X(n) for the service time of user n
- Service times X are exponentially distributed with parameter μ, so $E[X] = 1/\mu$
- Interarrival times and service times are independent
- We define $\rho = \lambda / \mu$, we shall see later how that relates to the ρ we considered when discussing Little's theorem
- Can we make use of the very special properties of Poisson processes to describe probabilistically the behavior of the system?

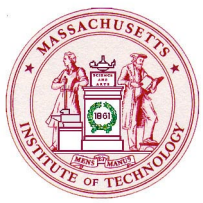


Derivation of occupancy distribution

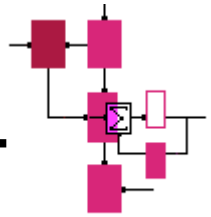


- Let us make use of the small δ approximation
- In a small δ interval, we have probability roughly $\lambda\delta$ of having an arrival and 1 - $\lambda\delta$ of having no arrival
- If there is a customer in the system, then we have probability roughly $\mu\delta$ of having a departure and 1 - $\mu\delta$ of having no departure
- Markov chain:state is number in the system

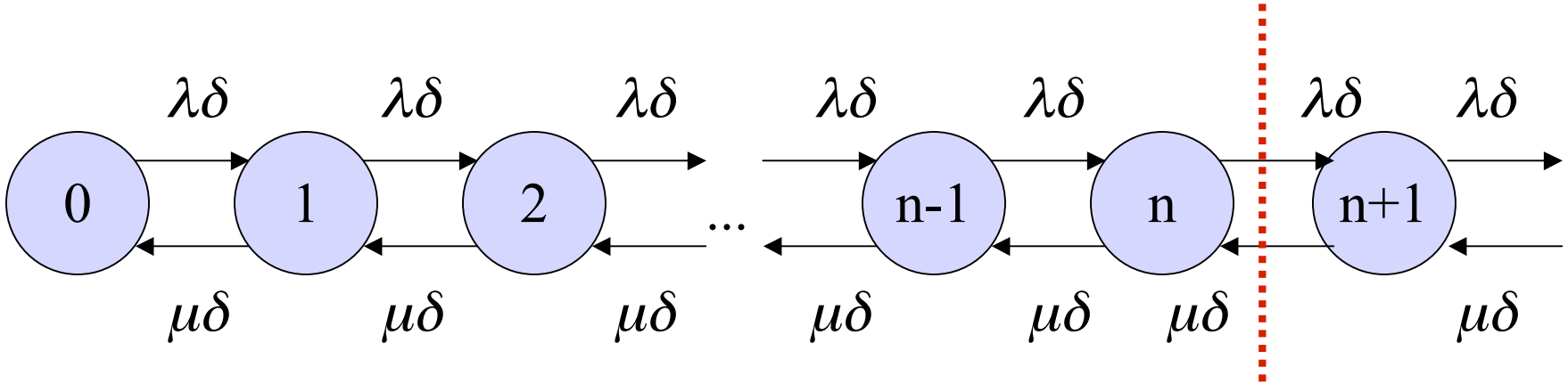




Markov chain for M/M/1

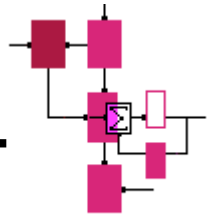


- In steady state, across some cut between two states, the proportion number of transitions from left to right must be the same as the proportion of transitions from right to left

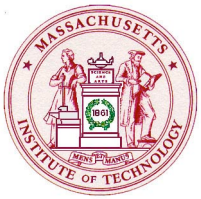


- Local balance equations, consider steady-state and replace N(t) with N

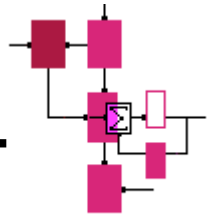
Balance equations



- We know that
- Let us use this fact to determine all the other probabilities
- We have
- Let us answer the second question:
  - we use the fact that Poisson arrivals see time average (PASTA)
  - the probability of having a random customer wait is ρ

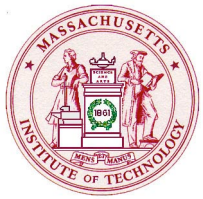


Mean values

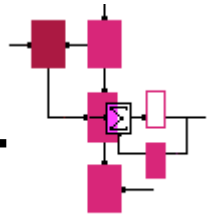


- We can now make use of Little's theorem to answer our first set of questions:

- What is the wait in queue, ? Use independence of service times to get = - $1/\mu$

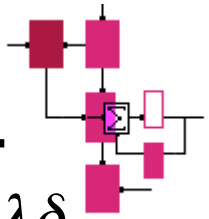


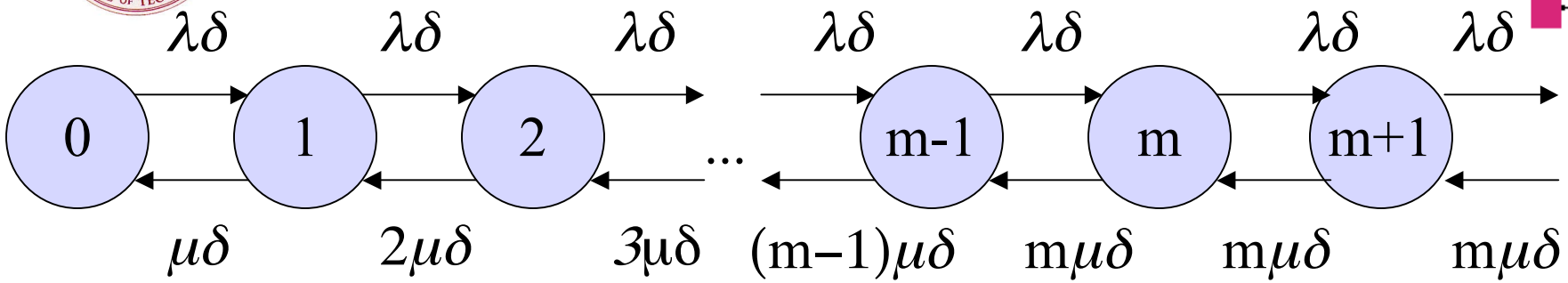
More queue scenarios

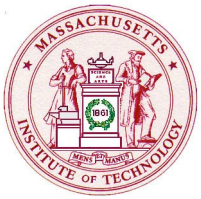


- A similar type of analysis holds for other queue scenarios:
  - set up a Markov chain
  - determine balance equations
  - use the fact that all probabilities sum to 1
  - derive everything else from there
- M/M/m queue: Poisson arrivals, exponential distribution of service time, m servers
- Similar analysis to before, except now the probability of a departure is proportional to the number of servers in use, because a departure occurs if AT LEAST one of the servers has a departure
- Now $\rho = m\mu$ – union bound holds for first order

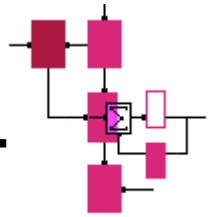
Markov chain for M/M/m





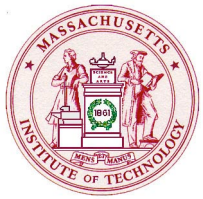


Let us answer our first two questions

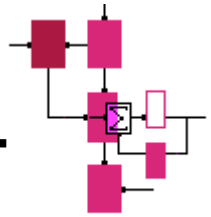


- Second question, what is the probability that a customer must wait in queue:Erlang C formula

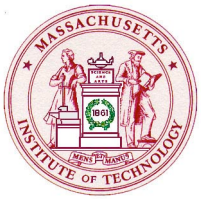
- Applying Little's theorem:



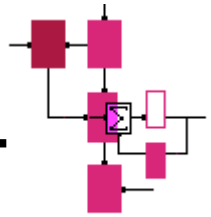
One server or many?



- We now have the tools to answer our third question: would I rather have a single more powerful server or many weaker servers?
- Would we rather have a single server with service rate mμ or m servers with service rate μ? Related question: if we have some channel do we want to allow statistical multiplexing onto m equal subchannels or not?
- For light loads, the delay is roughly m times for the system with several server channels
- For heavy loads, the delays are about the same

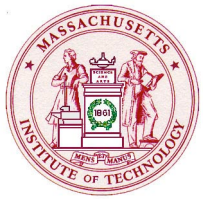


M/M/∞

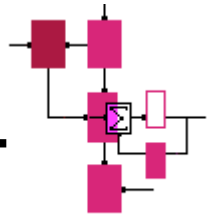


- Infinite number of servers
- Taking m to go to ∞ in the M/M/m system, we have that the occupancy distribution is Poisson with parameter λ/μ

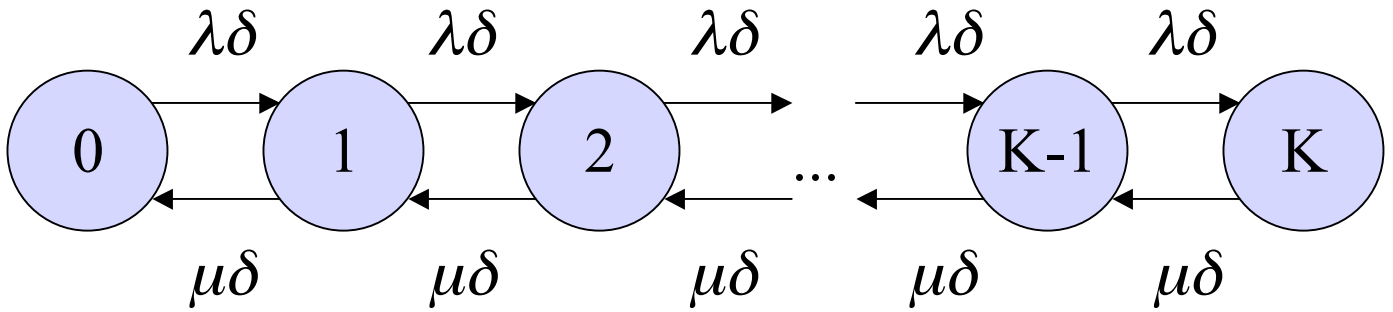
- $= 1/\mu$



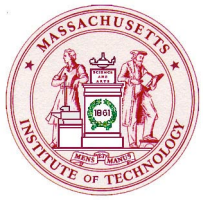
Finite-storage: M/M/1/K



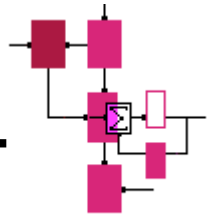
- Fourth question



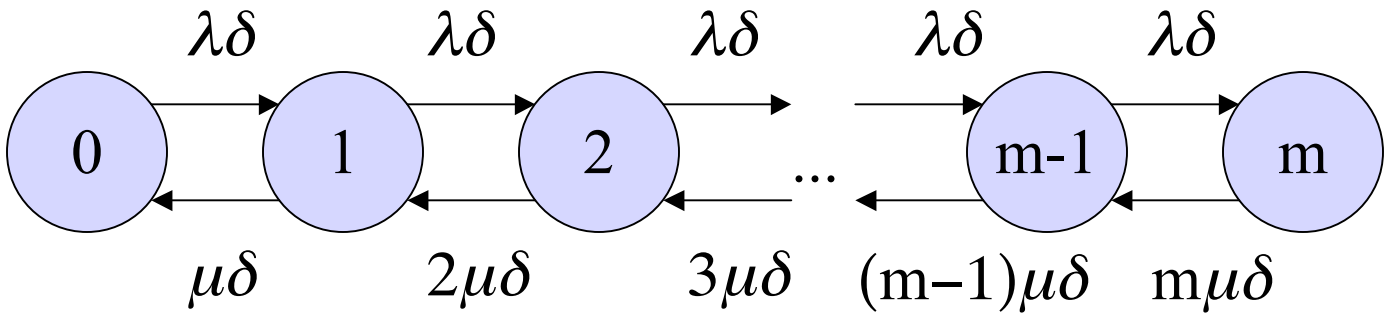
- PASTA yields probability that a new call is blocked



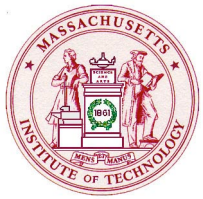
Another example of finite storage: M/M/m/m



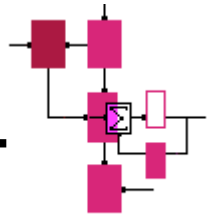
- Limited number of servers, no place to queue



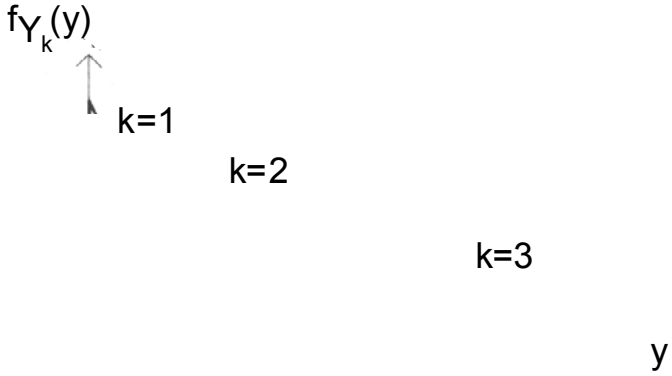
- Answer to fourth question is P(N=m), called Erlang B formula



Different distributions

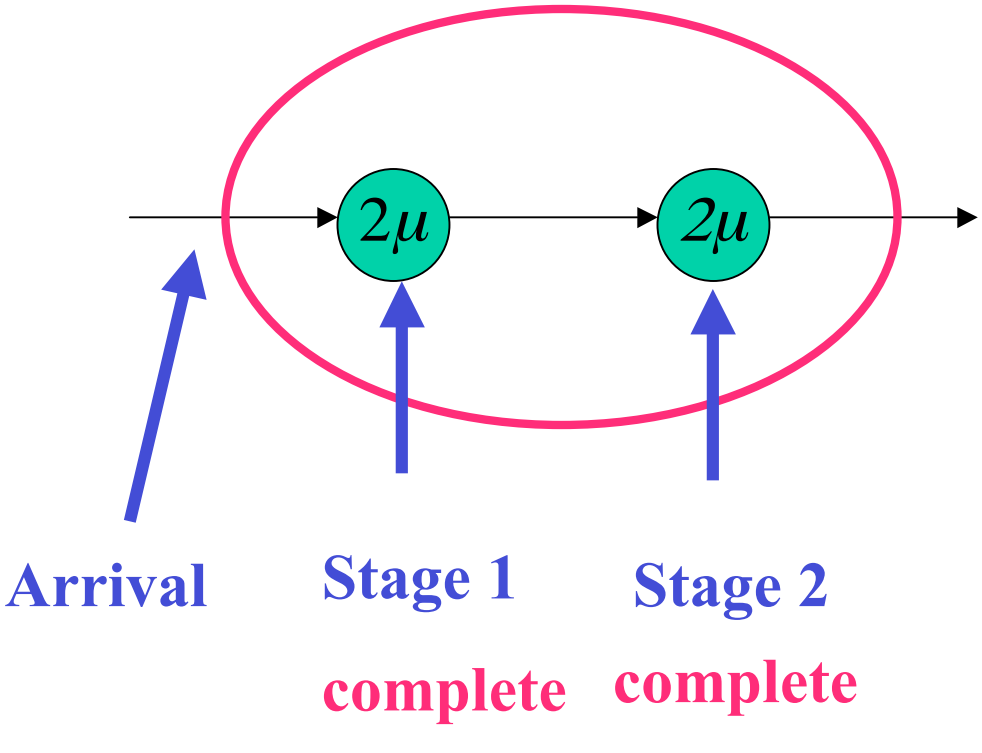


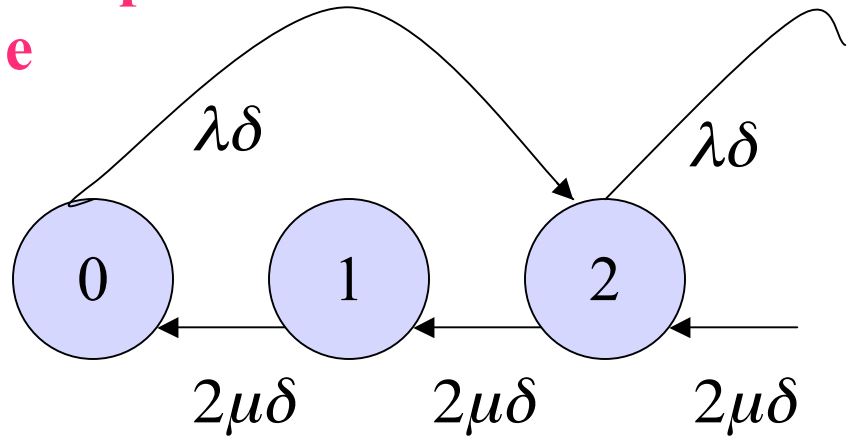
- Suppose we want to consider arrival and departure process that are not exponential
- Simplest extension: bulk arrivals
- We can easily generalize the M/M approach to Erlang distributions - this is the *method of stages*
- We can consider that we need k stages to establish a single arrival or departure
- We attempt to match the distribution to the closest Erlang, E_k

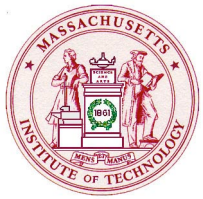


Method of stages

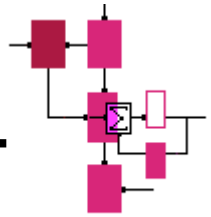
- Suppose we have a server with average service time $1/\mu$, and with second-order Erlang distribution (E_2) for its service time



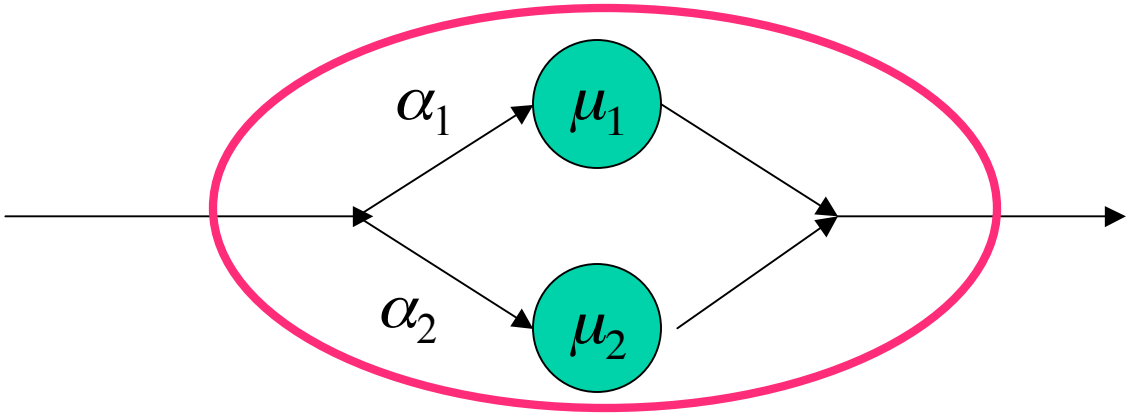




Parallel stages

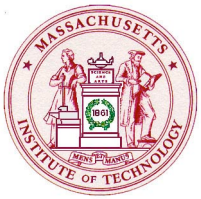


- Instead of arrival in serial stages, we can have parallel stages

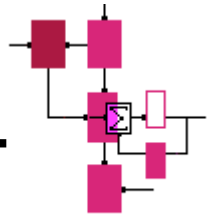


- Yields Hyperexponential distribution

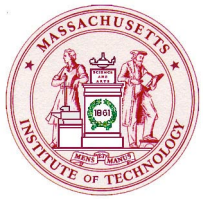
- The sum of the αs is 1



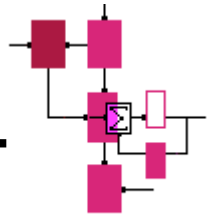
Beyond extensions to M/M/...



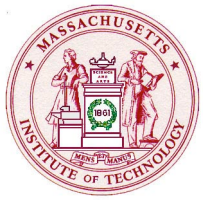
- The typicality of the state seen by an arrival is predicated on PASTA, so the M/ part is very powerful
- The assumption of memoryless arrivals is more commonly valid than the assumption of exponentially distributed service
- Extensions of the M/ ?/ using variations on the memoryless distribution become increasingly complicated
- More advanced approach: M/G/1 queues



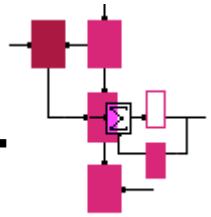
Transform review



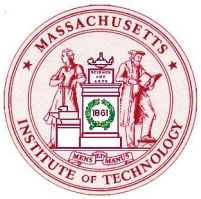
- Transform is a function of a parameter z or s
- Transform is well-defined for
- We define the z-transform for discrete random variable V and the Laplace transform for the continuous random variable B as:



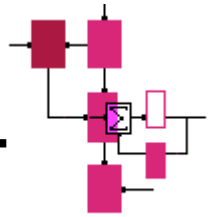
Transform review



- Properties of transforms:



Transform review



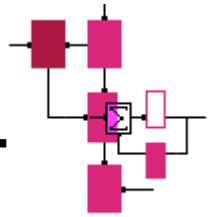
- Sums of independent random variables: the transform of the sum is the product of the transform
- Random sum of random variables:

  Let there be V IID random variables B_k

  the transform of $\sum_{k=0}^{V} B_k$ is $V(B^*(s))$

- Can derive compute transforms, manipulate in the transform domain, then derive distribution using inverse transform

Transform review



- Example: sum of independent Gaussian random variables is Gaussian

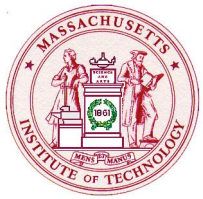
$$f_{X_i}(x) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{\frac{-(x-\mu_i)^2}{2\sigma_i^2}}$$

$$X_i^*(s) = E\left[e^{-sX_i}\right] = \int_x e^{-sx} f_{X_i}(x)dx = e^{\left(\frac{s^2\sigma_i^2}{2}\right)-s\mu_i}$$

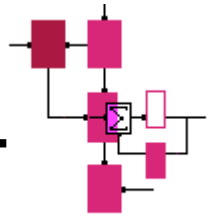
$$(X_1 + X_2)^*(s) = X_1^*(s).X_2^*(s)$$

$$= e^{\left(\frac{s^2\sigma_1^2}{2}\right)-s\mu_1}.e^{\left(\frac{s^2\sigma_2^2}{2}\right)-s\mu_2}$$

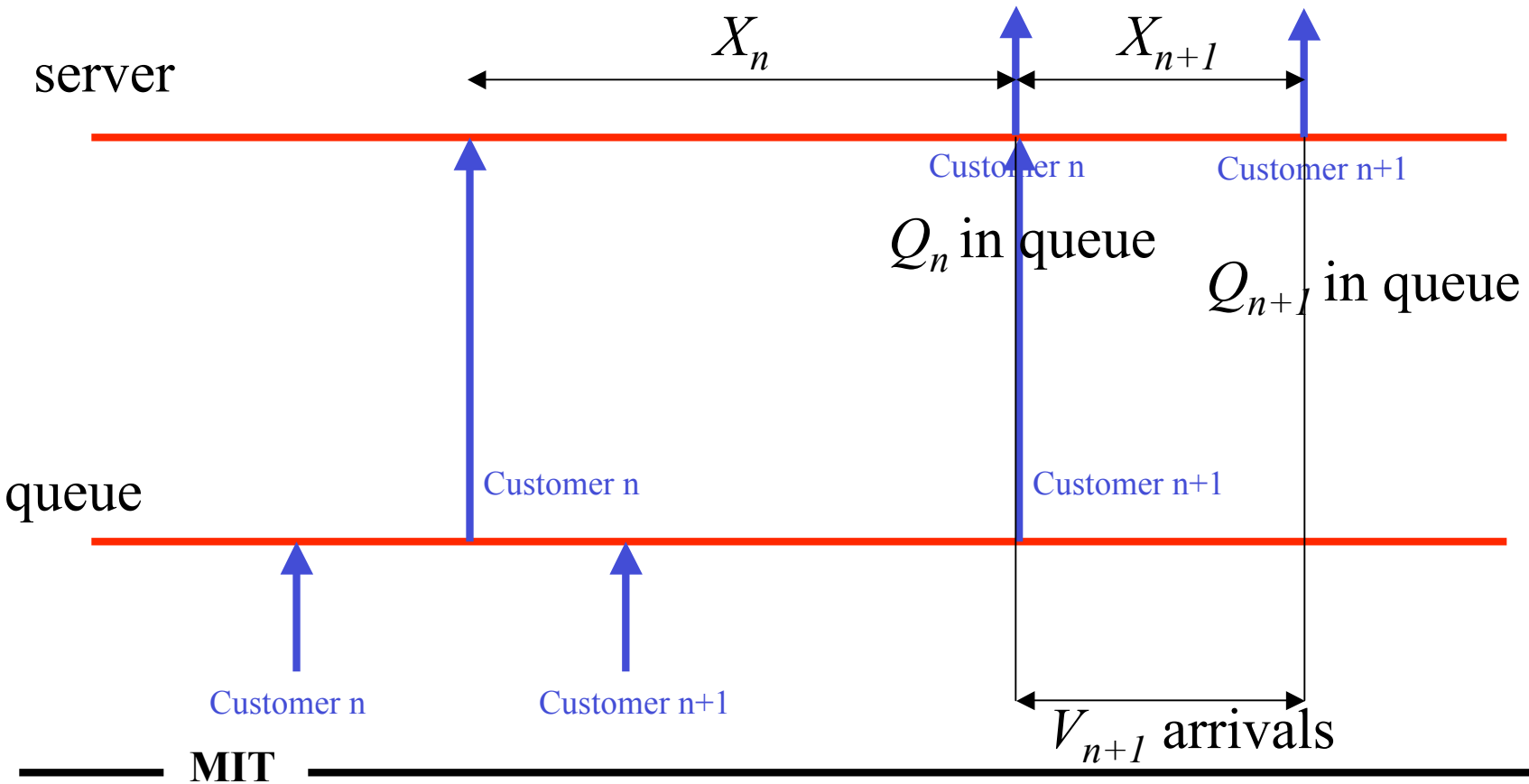
$$= e^{\left(\frac{s^2(\sigma_1^2+\sigma_2^2)}{2}\right)-s(\mu_1+\mu_2)}$$

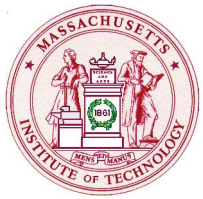


M/G/1 queue occupation

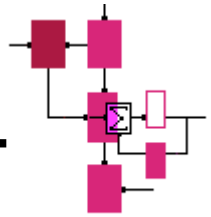


- Recall that PASTA holds
- Case when a departing customer leaves a non-empty queue

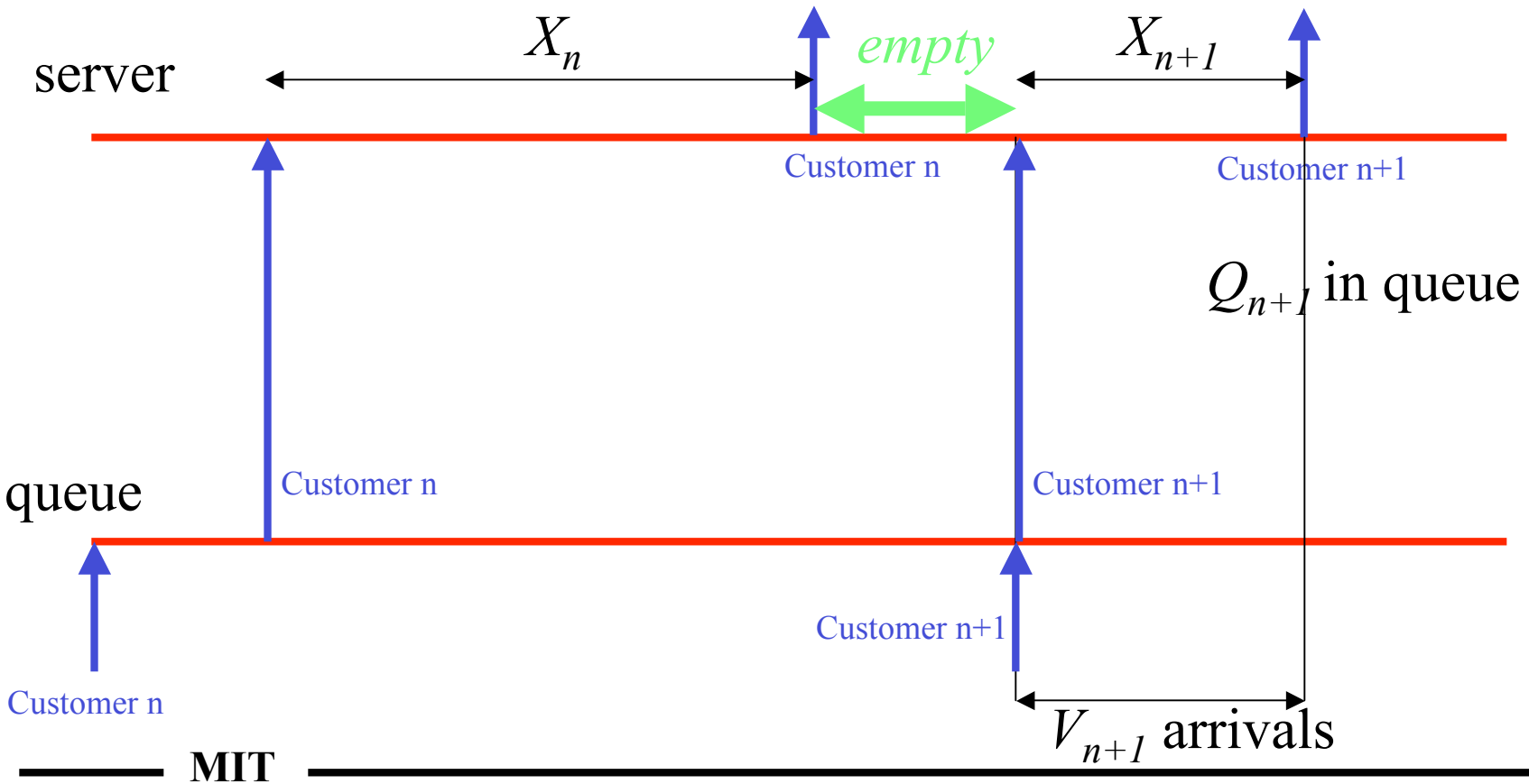




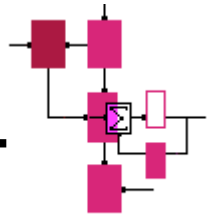
M/G/1 queue occupation



- Recall that PASTA holds
- Case when a departing customer leaves an empty queue



Difference equations



- We can express the events in a difference equation

$$Q_{n+1} = \begin{cases} Q_n - 1 + V_{n+1} & Q_n > 0 \\ V_{n+1} & Q_n = 0 \end{cases}$$

$$Q_{n+1} = Q_n - \Delta_{Q_n} + V_{n+1}$$

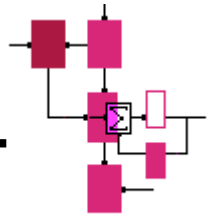
$$\Delta_k = \begin{cases} 1 & k > 0 \\ 0 & k_n = 0 \end{cases}$$

$$\Rightarrow E\left[z^{Q_{n+1}}\right] = E\left[z^{Q_n - \Delta_{Q_n} + V_{n+1}}\right]$$

$$Q_n(z) = E\left[z^{Q_n - \Delta_{Q_n} + V_{n+1}}\right] = E\left[z^{Q_n - \Delta_{Q_n}}\right] E\left[z^{V_{n+1}}\right]$$

because the number of arrivals is independent of the past (Poisson)

Difference equations



- Use the steady-state assumptions

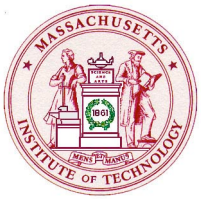
$$E\left[z^{V_{n+1}}\right] = V(z)$$

$$\Rightarrow Q_{n+1}(z) = E\left[z^{Q_n - \Delta_{Q_n}}\right] V(z)$$

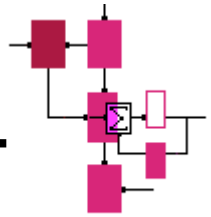
$$E\left[z^{Q_n - \Delta_{Q_n}}\right] = \sum_{k=0}^{\infty} z^{k-\Delta_k} P(Q_n = k)$$

$$= P(Q_n = 0) + \sum_{k=1}^{\infty} z^{k-1} P(Q_n = k)$$

$$= P(Q_n = 0) + \frac{1}{z}\sum_{k=0}^{\infty} z^{k} P(Q_n = k) - \frac{1}{z} P(Q_n = 0)$$



Difference equations



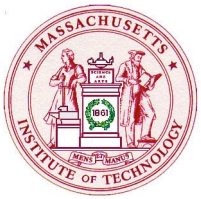
- Use the steady-state assumptions again

$$Q_{n+1}(z) = Q(z)$$

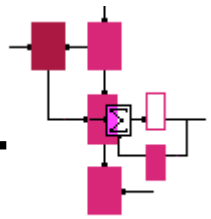
$$\Rightarrow Q(z) = V(z)\left(P(Q=0) + \frac{Q(z) - P(Q=0)}{z} \right)$$

$$\Rightarrow Q(z) = V(z) \frac{(1-\rho)\left(1 - \frac{1}{z}\right)}{1 - \frac{V(z)}{z}}$$

- How do we relate this to the service time?



Service time and V



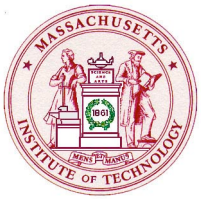
- Again use PASTA

$$V(z) = \int_0^\infty e^{-\lambda x} \left(\sum_{k=0}^{\infty} \frac{(\lambda x z)^k}{k!} \right) f_X(x)\,dx$$

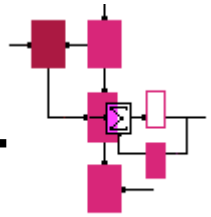
$$= \int_0^\infty e^{-\lambda x} e^{\lambda x z} f_X(x)\,dx$$

$$= \int_0^\infty e^{-(\lambda - \lambda z)x} f_X(x)\,dx$$

$$= X^*(\lambda - \lambda z)$$



P-K transform equation



- P-K transform equation

$$Q(z) = X^*(\lambda - \lambda z)\frac{(1-\rho)(1-z)}{X^*(\lambda - \lambda z) - z}$$

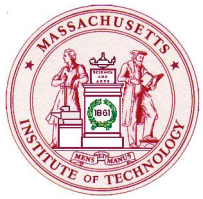
- Check: M/M/1

$$X^*(s) = \frac{\mu}{\mu + s}$$

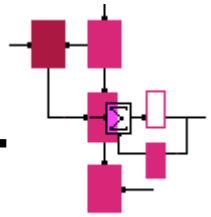
$$Q(z) = \frac{\mu}{\mu + \lambda - \lambda z}\frac{(1-\rho)(1-z)}{\left(\frac{\mu}{\mu + \lambda - \lambda z}\right) - z} = \frac{1-\rho}{1-\rho z}$$

$$\Rightarrow P(Q = k) = (1-\rho)\rho^k$$

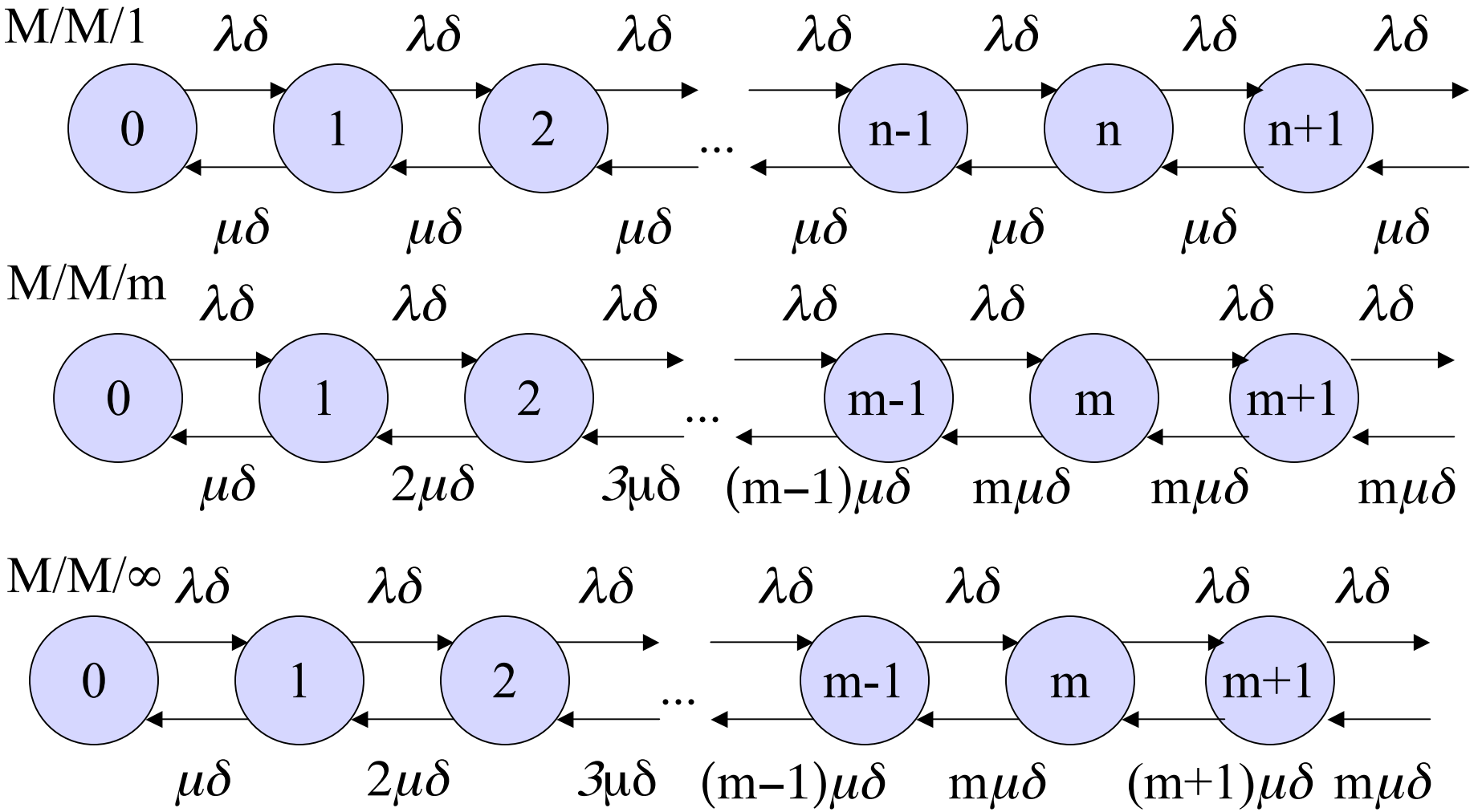
same as for N!

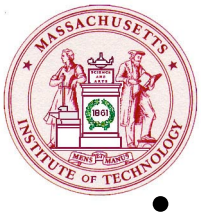


M/M/x queues

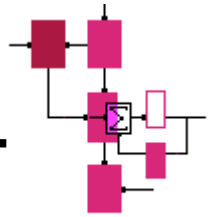


- Markov chains:





Steady-state distribution



- Use **Markov** property: denote in steady-state

$$p_i = P(N = i), P_{j,i} = P(N(k+1) = j \mid N(k) = i)$$

$$P\left(N(m) = j \mid N(m+1) = i, N(m+2) = i_2, \ldots, N(m+k) = i_k\right)$$

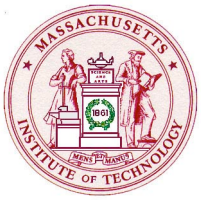
$$= \frac{P\left(N(m) = j, N(m+1) = i, N(m+2) = i_2, \ldots, N(m+k) = i_k\right)}{P\left(N(m+1) = i, N(m+2) = i_2, \ldots, N(m+k) = i_k\right)}$$

$$= \frac{P\left(N(m) = j, N(m+1) = i\right) P\left(N(m+2) = i_2, \ldots, N(m+k) = i_k \mid N(m) = j, N(m+1) = i\right)}{P\left(N(m+1) = i\right) P\left(N(m+2) = i_2, \ldots, N(m+k) = i_k \mid N(m+1) = i\right)}$$

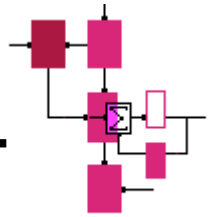
$$= \frac{P\left(N(m) = j, N(m+1) = i\right)}{P\left(N(m+1) = i\right)}$$

$$= \frac{P\left(N(m) = j\right) P\left(N(m+1) = i \mid N(m) = j\right)}{P\left(N(m+1) = i\right)}$$

$$= \frac{p_i P_{j,i}}{p_j}$$



Reverse chain



- Let us consider a chain where the transition probabilities are:

$$P^*_{i,j} = \frac{p_j P_{j,i}}{p_i}$$

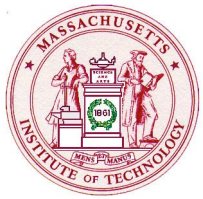
- Steady-state probabilities:

$$p^*_i = \sum_{j=0}^{\infty} p^*_j P^*_{j,i} = \sum_{j=0}^{\infty} p^*_j \frac{p_i P_{i,j}}{p_j} = p_i \sum_{j=0}^{\infty} p^*_j \frac{P_{i,j}}{p_j}$$

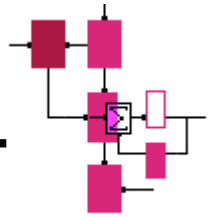
$$p_i = \sum_{j=0}^{\infty} p_j P_{j,i} = \sum_{j=0}^{\infty} p_j \frac{p_i}{p_j} P^*_{i,j} = p_i \sum_{j=0}^{\infty} P^*_{i,j} \Rightarrow \sum_{j=0}^{\infty} P^*_{i,j} = 1$$

- Hence,

$$p_i = p^*_i$$

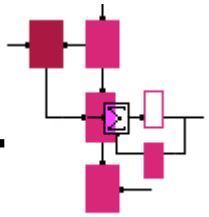


Burke's theorem and reversibility

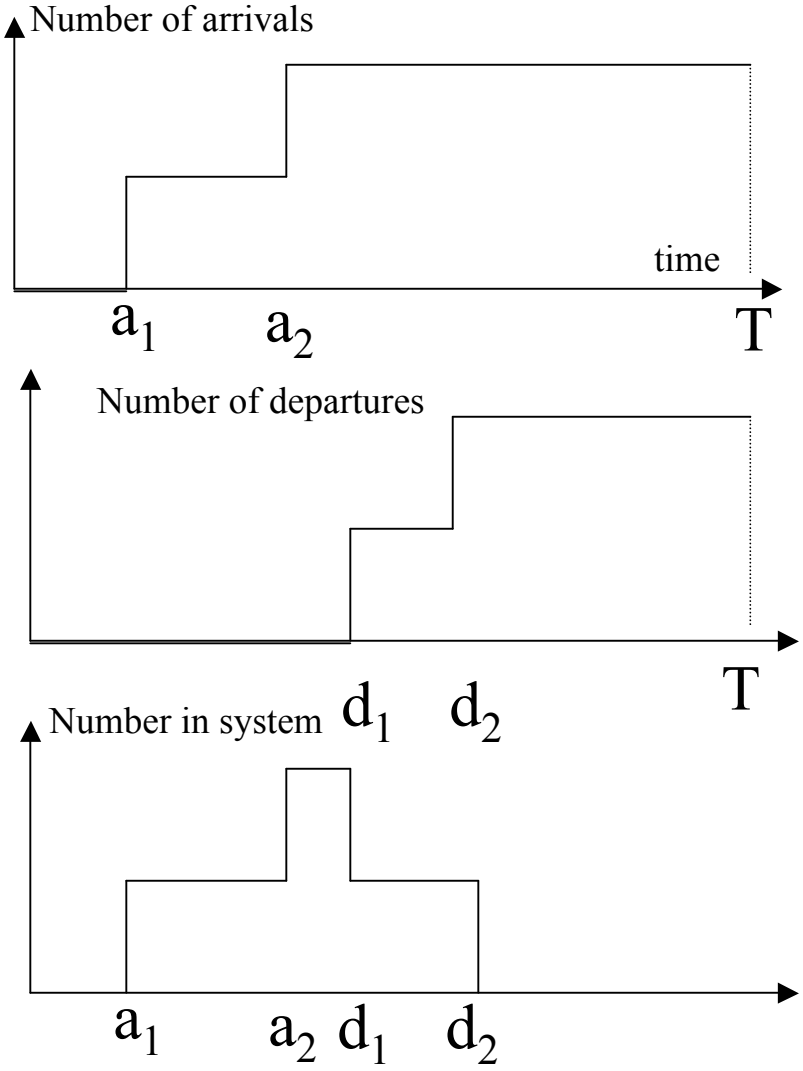


- Burke's theorem: consider an M/M/1, M/M/m or M/M/∞ system with arrival rate λ. Suppose that the system starts in steady-state. Then:
  - The departure process is Poisson with rate λ
  - At each time t, the number of customers in the system is independent of the sequence of departures prior to t
- Proof:
  - For the first part of the theorem, note that the forward and reversed systems are statistically indistinguishable in steady-state
  - The departures prior to t in the forward process are also the arrivals after t in the reversed system and the arrival process in the reversed system is independent Poisson

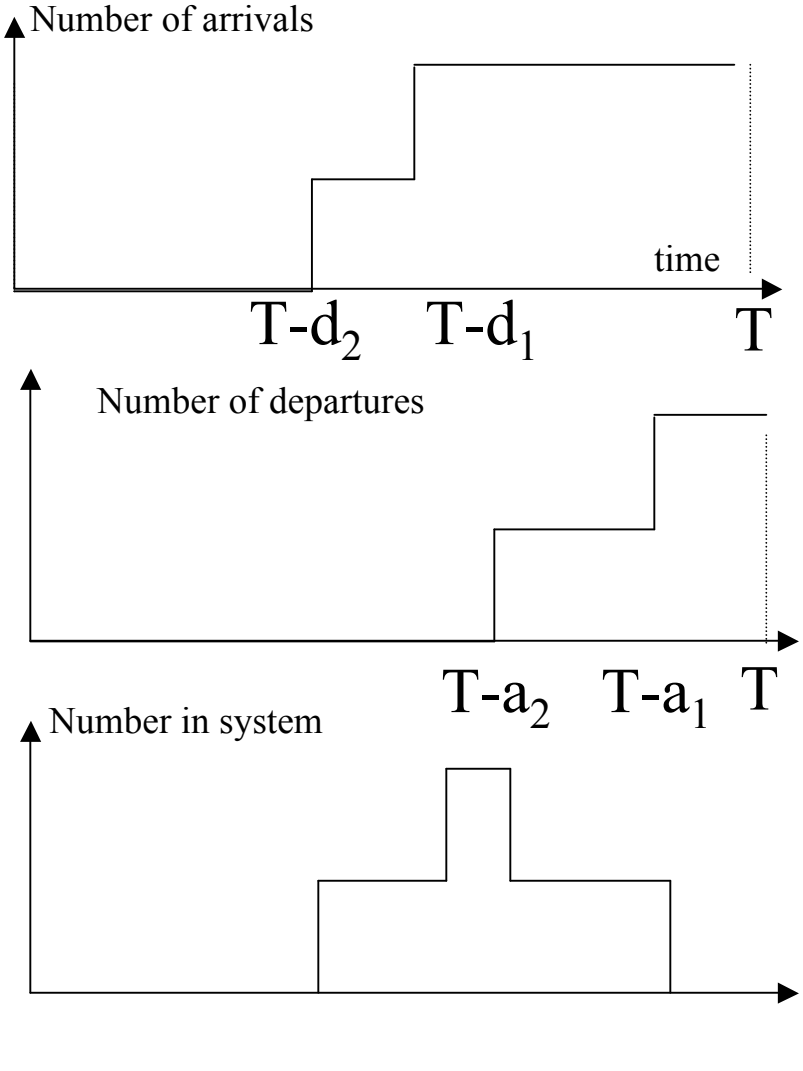
Burke's theorem

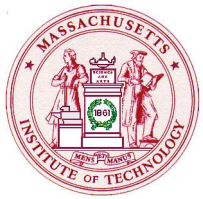


Forward system

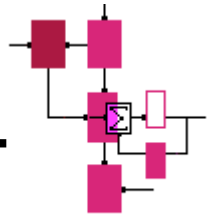


Reversed system

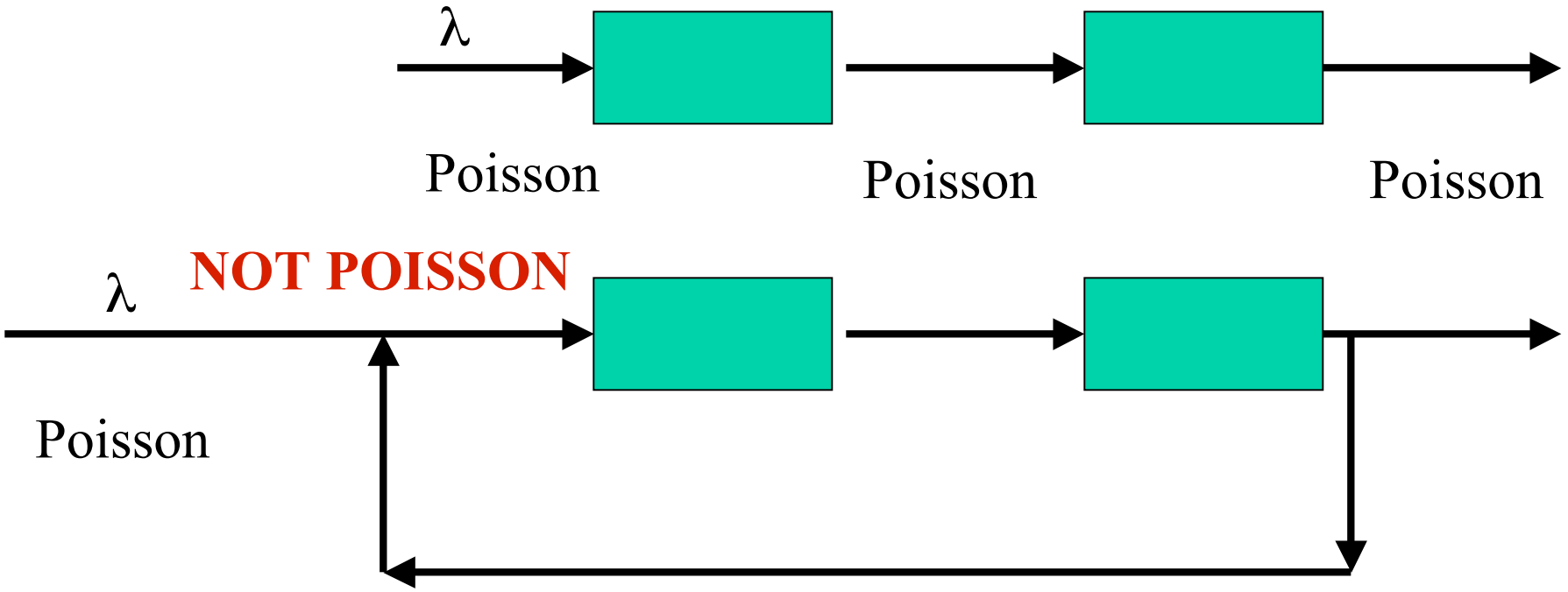


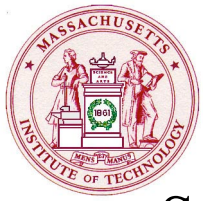


Networks of queues

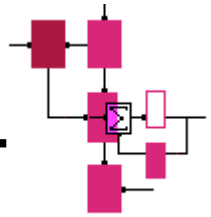


- Closed form solutions are difficult to obtain
- Poisson with feedback does not remain Poisson





Network of queues

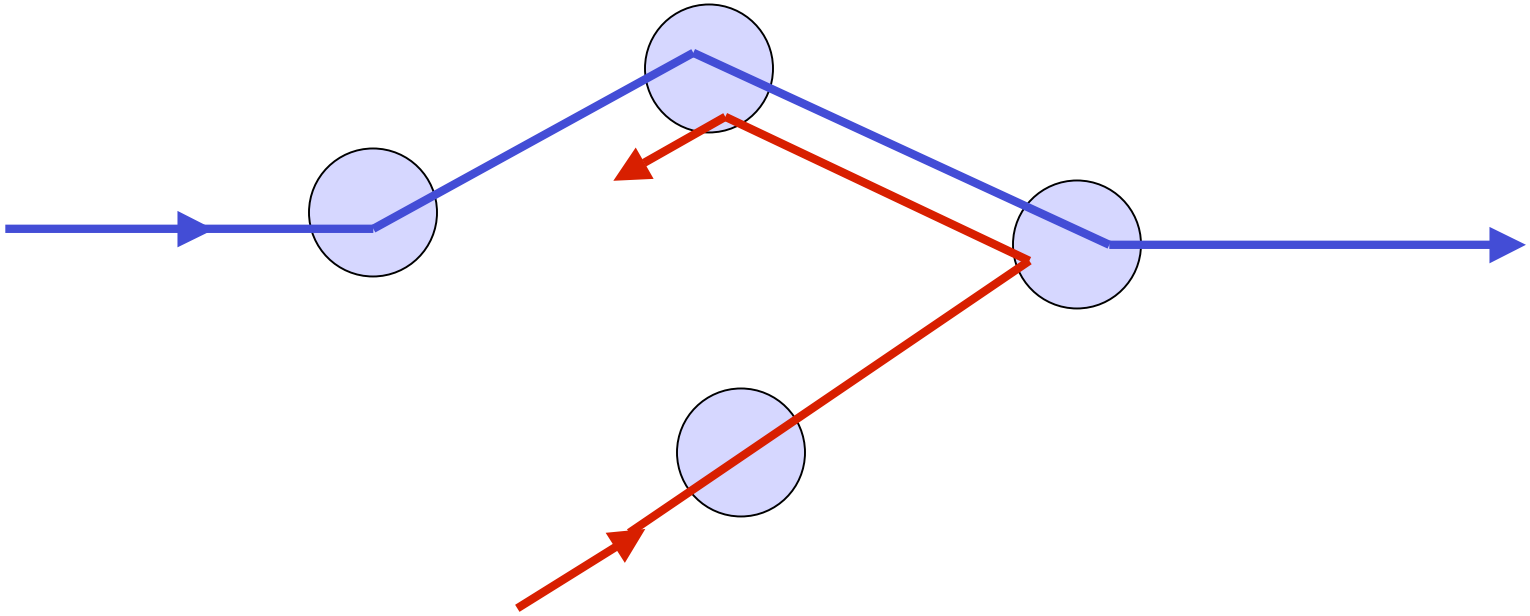


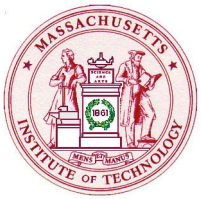
- Several streams, each on a path p, each with rate $\lambda(p)$
- Let us look at directed link (i,j):

$$\lambda(i,j) = \sum_{\text{all paths p traversing link (i,j)}} \lambda(p)$$

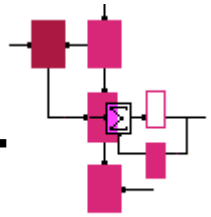
$\mu(i,j)$ = service rate on link (i, j)

N(i, j) = average number of packets on link (i, j)





Kleinrock independence assumption



- Assume all queues behave like M/M/1 with arrival rate $\lambda(i,j)$, service rate $\mu(i,j)$, and service/propagation delay $d(i,j)$
- Then

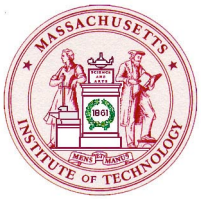
$$N_{i,j} = \frac{\lambda_{i,j}}{\mu_{i,j} - \lambda_{i,j}} + \lambda_{i,j} d_{i,j}$$

average number of packets in the whole network

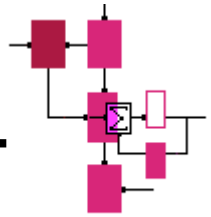
$$N = \sum_{i,j} N_{i,j}$$

average time in the system (using Little's theorem)

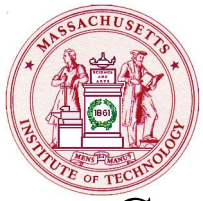
$$T = \frac{N}{\sum_{p} \lambda_p}$$



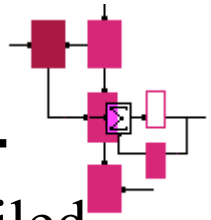
How good is it?



- Good for densely connected networks and moderate to heavy loads
- Good to guide topology design before involving simulation, other applications where a rough estimate is needed
- Other methods involve G/G/m approximations, using first and second moments of interarrival and service times
- Are there any networks of queues where we can establish analytical results?



Reversibility in Markov queues



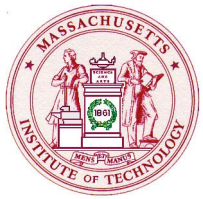
- Consider a reversed chain: reversibility exists iff the detailed balance equations hold
- Reversed chain :
  - Reversed chain is irreducible, aperiodic and has the same stationary distribution as the forward chain
  - If we can find positive p_i numbers that sum to 1 such that

$$P^*_{i,j} = \frac{p_j P_{j,i}}{p_i} \text{ satisfy } \sum_j P^*_{i,j} = 1 \text{ then}$$

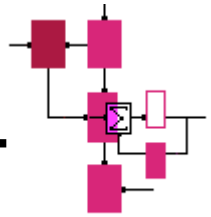
the p_is are the stationary distributions and the

$P^*_{i,j}$ s are the transition probabilities of the reversed chain

$$\text{Note}: \sum_j p_j P_{j,i} = p_i \sum_j P^*_{i,j} = p_i$$

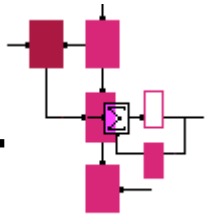


Jackson's theorem

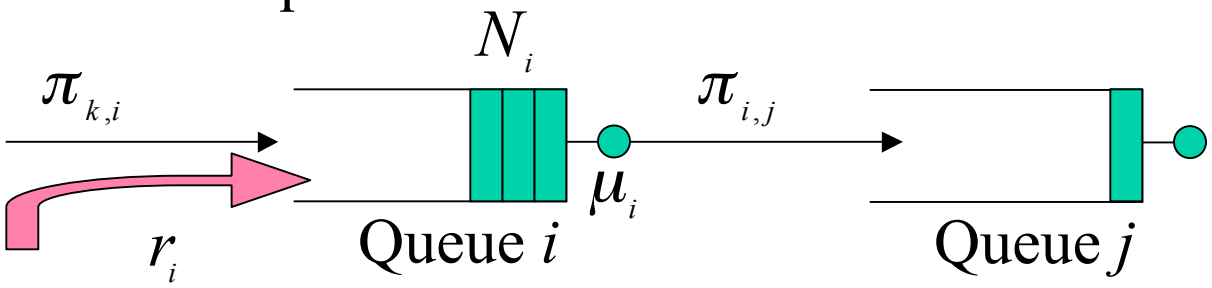


- Assuming that:
  - arrival processes from outside the network are Poisson
  - at each queue, streams have the same exponential service time distribution and a single server
  - interarrival times and service times are independent
- Then:
  - the steady state occupancy probabilities in each queue are the same as if the queue were M/M/1 in isolation
- These results can be extended to:
  - state-dependent service rates (for instance M/M/m queues)
  - closed networks

Jackson's Theorem



- Model: *K* queues

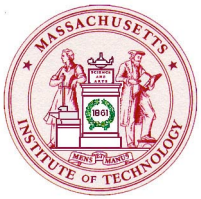


$$\lambda_j = r_j + \sum_{i=1}^{\infty} \lambda_i \pi_{i,j} \qquad \rho_j = \frac{\lambda_j}{\mu_j}$$

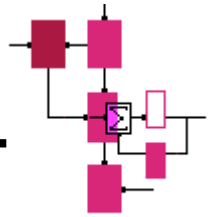
- Jackson's Theorem:

$$P(N_i = n_i) = \rho_i^{n_i}(1 - \rho_i) \text{ and}$$

$$P(\vec{N} = \vec{n}) = \prod_{i=1}^{K} P(N_i = n_i)$$



Jackson's Theorem



- Some definitions:

$$P^*_{\vec{n},\vec{n}'} = \frac{P\left(\vec{N} = \vec{n}'\right) P_{\vec{n}',\vec{n}}}{P\left(\vec{N} = \vec{n}\right)}$$

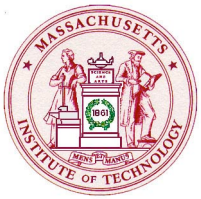
$$\vec{n}_{j^+} \stackrel{\Delta}{=} \left(n_1, n_2, \ldots, n_{j-1}, n_j + 1, n_{j+1}, \ldots, n_K\right)$$

$$\vec{n}_{j^-} \stackrel{\Delta}{=} \left(n_1, n_2, \ldots, n_{j-1}, n_j - 1, n_{j+1}, \ldots, n_K\right)$$

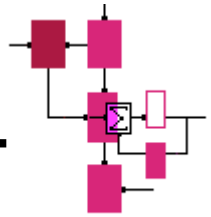
Exogenous

$$\vec{n}_{i^+,j^-} \stackrel{\Delta}{=} \left(n_1, n_2, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_{j-1}, n_j + 1, n_{j+1}, \ldots, n_K\right)$$

Endogenous



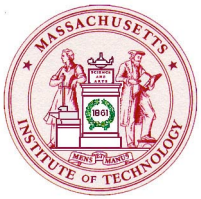
Jackson's Theorem-Proof Technique



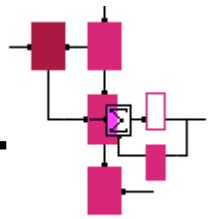
- Establishing the balance equations is difficult because of different types of transitions
- We assume the answer
- Using this answer, we see whether the forward and reverse system check out
- From the statement of Jackson's Theorem, we have that:

$$P\left(\vec{N}=\vec{n}\right)\rho_j = P\left(\vec{N}=\vec{n}^{\,j^+}\right)$$

$$P\left(\vec{N}=\vec{n}\right)\rho_i = \rho_j P\left(\vec{N}=\vec{n}^{\,i^+,j^-}\right)$$



Exogenous arrivals/departures



- Transition probabilities for both chains

$$P_{\vec{n},\vec{n}_{j^+}} = r_j \delta$$

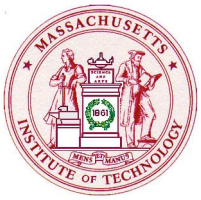
$$P_{\vec{n},\vec{n}_{j^-}} = \mu_j \left(1 - \sum_{i=1}^{K} \pi_{j,i}\right)$$

By our assumption :

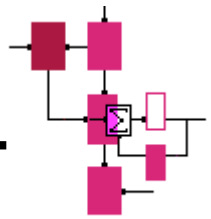
$$P\left(\vec{N} = \vec{n}\right) o_j = P\left(\vec{N} = \vec{n}^{j^+}\right)$$

$$P^*_{\vec{n},\vec{n}_{j^+}} = \lambda_j \left(1 - \sum_{i=1}^{K} \pi_{j,i}\right)$$

$$P^*_{\vec{n},\vec{n}_{j^-}} = r_j \delta \frac{\mu_j}{\lambda_j}$$



Endogenous arrivals/departures



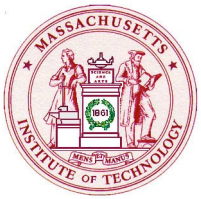
- Transition probabilities for both chains

$$P_{\vec{n},\vec{n}_{i^+,j^-}} = \mu_j \pi_{j,i} \delta$$

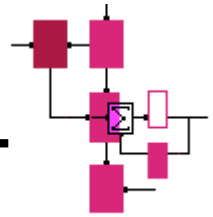
From our assumption :

$$P(\vec{N} = \vec{n}) \rho_i = \rho_j P(\vec{N} = \vec{n}^{i^+,j^-})$$

$$P^*_{\vec{n}_{i^+,j^-},\vec{n}} = \delta \frac{\mu_i \lambda_j}{\lambda_i} \pi_{j,i}$$



Check conditions for stationary distributions

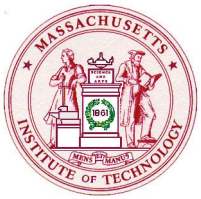


- Look at forward chain:

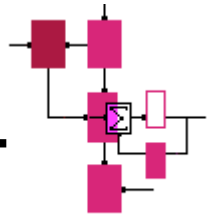
$$\sum_{\vec{n}'} P_{\vec{n},\vec{n}'} = \sum_{j=1}^{K} P_{\vec{n},\vec{n}_{j^+}} + \sum_{i,j|n_j>0} P_{\vec{n},\vec{n}_{i^+,j^-}} + \sum_{j|n_j>0} P_{\vec{n},\vec{n}_{j^-}}$$

$$= \sum_{j=1}^{K} r_j \delta + \sum_{i,j|n_j>0} \mu_j \pi_{j,i} \delta + \sum_{j|n_j>0} \delta \mu_j \left(1 - \sum_{i=1}^{K} \pi_{j,i}\right)$$

$$= \sum_{j=1}^{K} r_j \delta + \sum_{j|n_j>0} \delta \mu_j$$



Check conditions for stationary distributions



- Look at reverse chain:

$$\sum_{\vec{n}'} P^*_{\vec{n},\vec{n}'} = \sum_{j=1}^{K} P^*_{\vec{n},\vec{n}_{j^+}} + \sum_{i,j|n_j>0} P^*_{\vec{n},\vec{n}_{i^+,j^-}} + \sum_{j|n_j>0} P^*_{\vec{n},\vec{n}_{j-}}$$

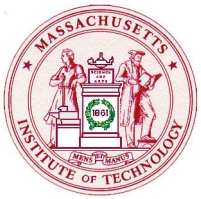
$$= \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} \pi_{j,i}\right)\delta + \sum_{i,j|n_j>0} \frac{\mu_j \lambda_i \pi_{i,j}}{\lambda_j}\delta + \sum_{j|n_j>0} \delta \frac{\mu_j}{\lambda_j} r_j$$

$$= \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} \pi_{j,i}\right)\delta + + \sum_{j|n_j>0} \delta \frac{\mu_j \left(r_j + \sum_{i=1}^{K} \lambda_i \pi_{i,j}\right)}{\lambda_j}$$

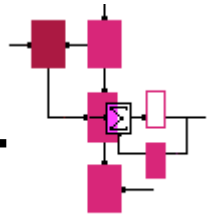
$$= \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{K} \pi_{j,i}\right)\delta + \sum_{j|n_j>0} \delta\mu_j$$

$$= \sum_{j=1}^{K} r_j\delta + \sum_{j|n_j>0} \delta\mu_j$$

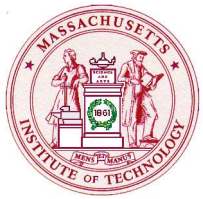
Same as for forward!



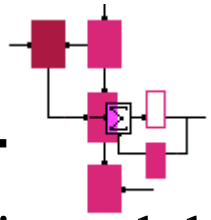
Extensions to Jackson's Theorem



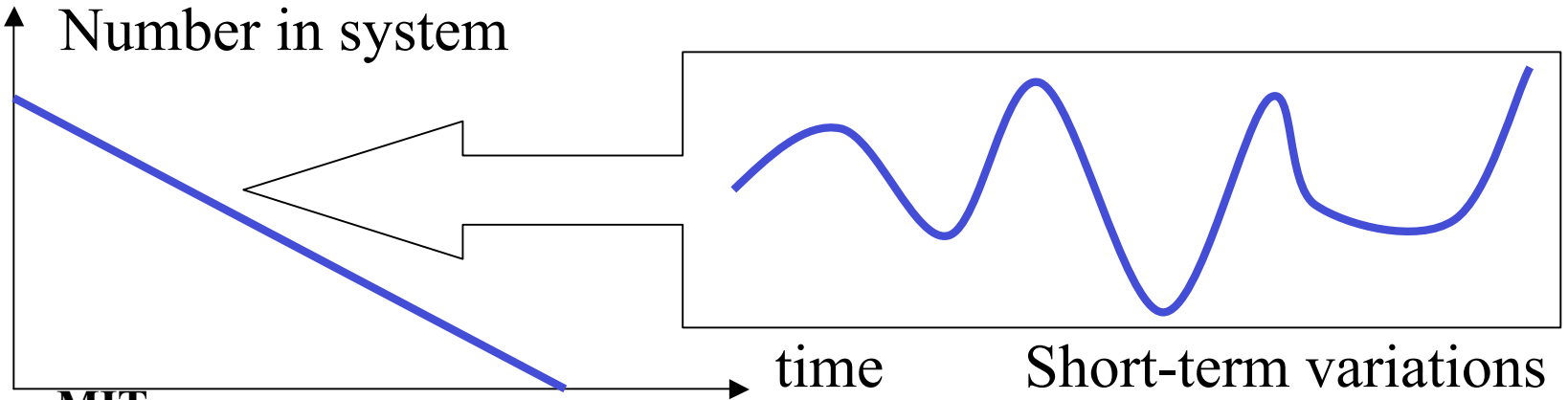
- Closed systems (only endogenous arrivals/departures)
- State-dependent service rates - powerful extension
- Multiple classes of customers (can be combined with state-dependent service rates)
- Can use methods of stages and Hyperexponential tricks as for extensions of the M/M/x models
- De facto makes the M/M/1 model a canonical model for understanding complex systems
- More complex systems (with specific policies, or controls) require more advanced methods



Fluid models



- We can look at fluid models in which we consider the flow in and the flow out of a system
- Idea: Imagine that we have the system very backed up, would it eventually empty?
- No longer look at the short-term behavior, but concentrate instead on the derivative of the long-term behavior
- Very good for systems with control of queues
- Problem: addresses stability, delay but not short-term variations



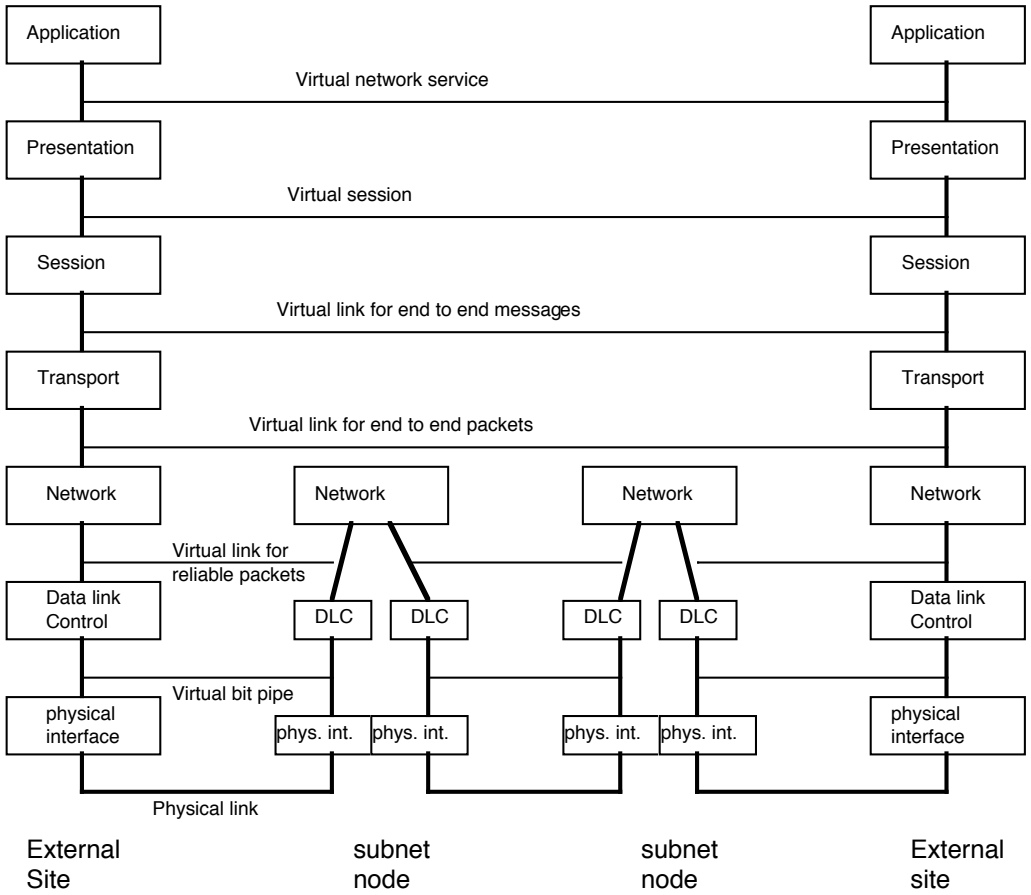
Higher Layer Protocols
TCP/IP and ATM

Eytan Modiano
Massachusetts Institute of Technology
Laboratory for Information and Decision Systems

Outline

- Network Layer and Internetworking
- The TCP/IP protocol suit
- ATM
- MPLS

Higher Layers



TCP, UDP

IP, ATM

The TCP/IP Protocol Suite

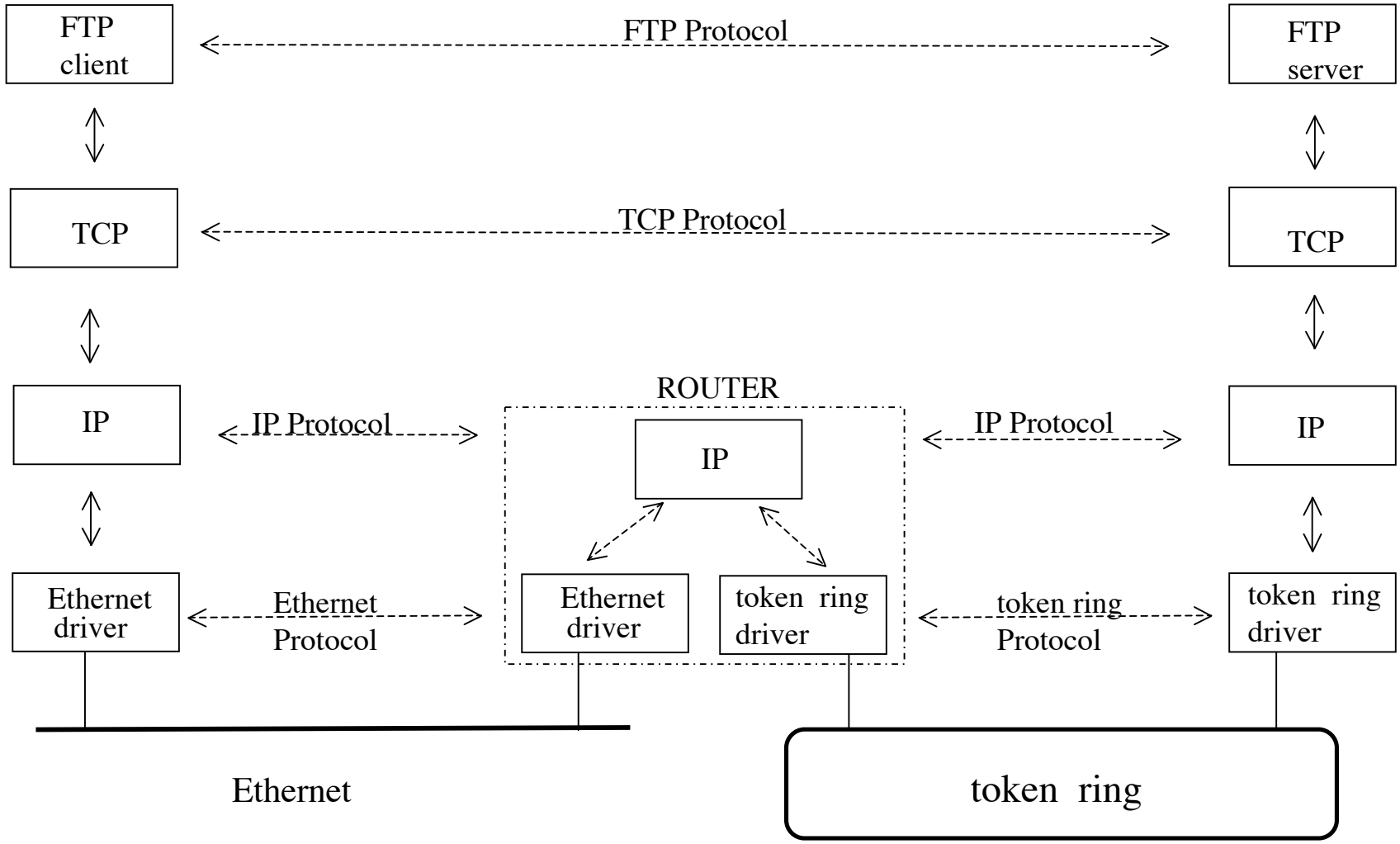
- Transmission Control Protocol / Internet Protocol
- Developed by DARPA to connect Universities and Research Labs

Four Layer model

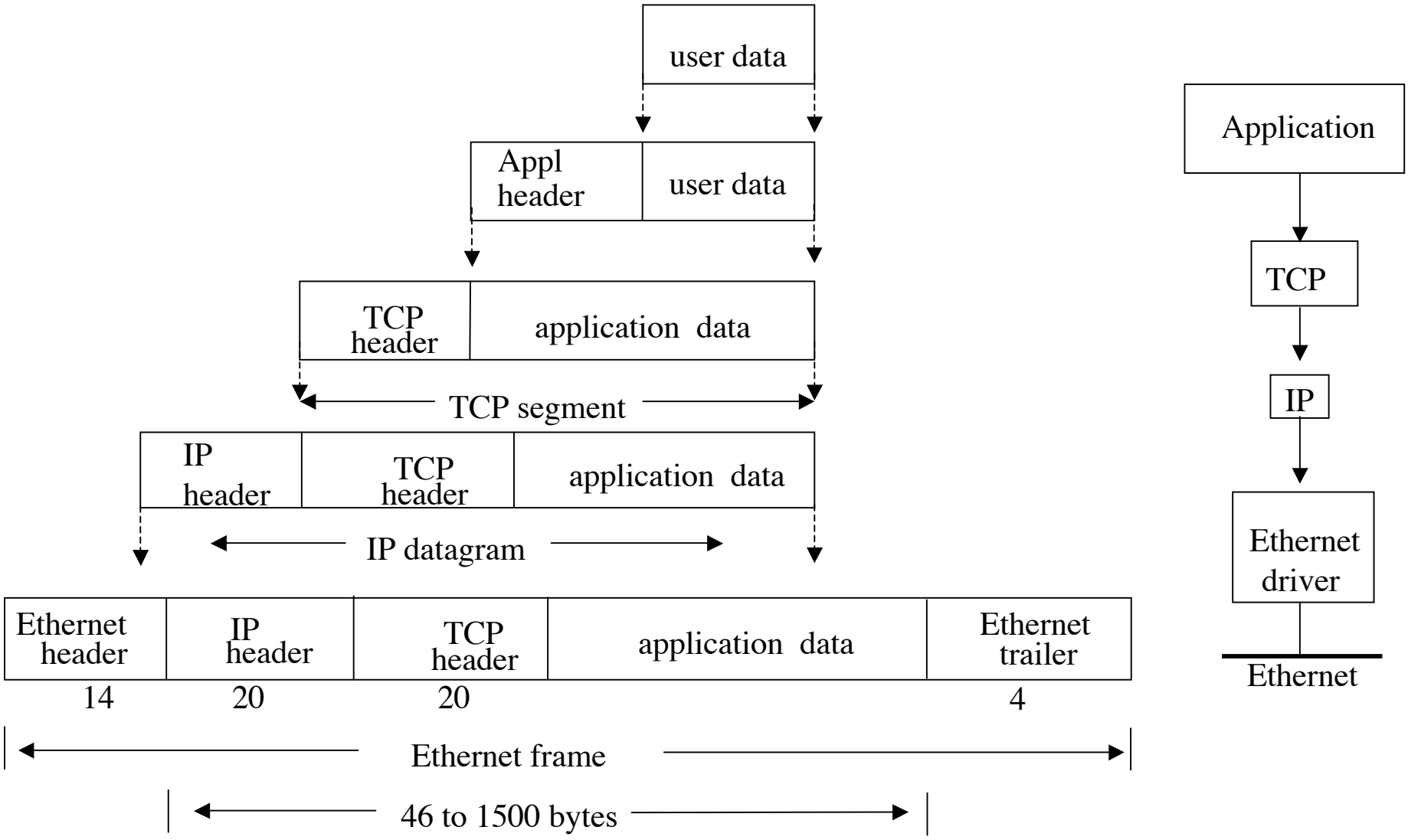
Applications	Telnet, FTP, email, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	Device drivers, interface cards

TCP - Transmission Control Protocol
UDP - User Datagram Protocol
IP - Internet Protocol

Internetworking with TCP/IP



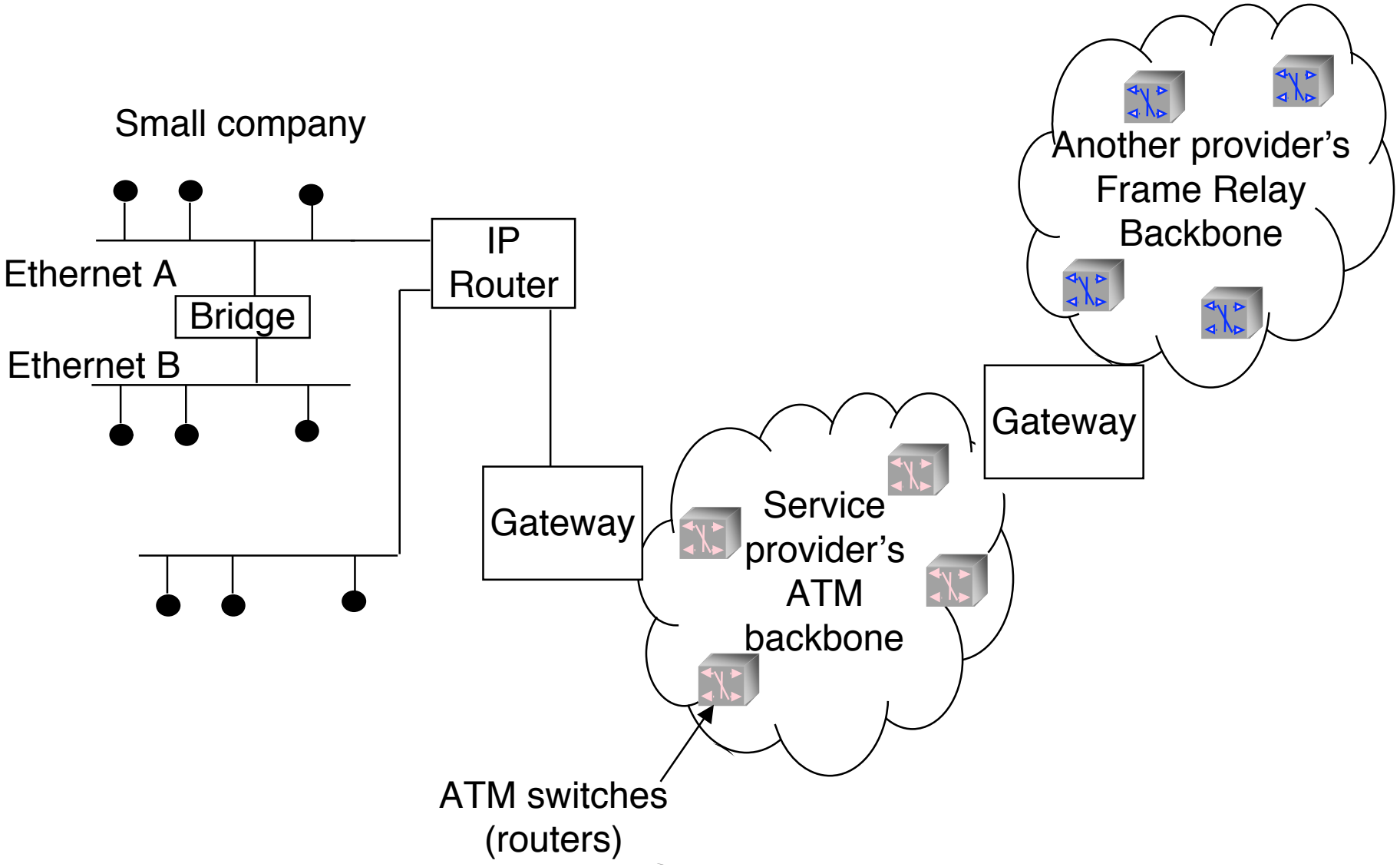
Encapsulation



Bridges, Routers and Gateways

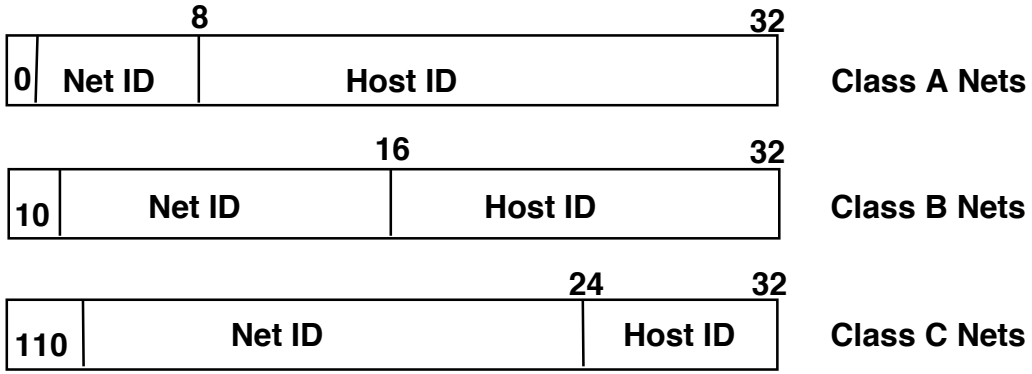
- A Bridge is used to connect multiple LAN segments
  - Layer 2 routing (Ethernet)
  - Does not know IP address
  - Varying levels of sophistication
    - Simple bridges just forward packets
    - smart bridges start looking like routers
- A Router is used to route connect between different networks using network layer address
  - Within or between Autonomous Systems
  - Using same protocol (e.g., IP, ATM)
- A Gateway connects between networks using different protocols
  - Protocol conversion
  - Address resolution
- These definitions are often mixed and seem to evolve!

Bridges, routers and gateways



IP addresses

- 32 bit address written as four decimal numbers
  - One per byte of address (e.g., 155.34.60.112)
- Hierarchical address structure
  - Network ID/ Host ID/ Port ID
  - Complete address called a socket
  - Network and host ID carried in IP Header
  - Port ID (sending process) carried in TCP header
- IP Address classes:



Class D is for multicast traffic

Host Names

- Each machine also has a unique name
- Domain name System: A distributed database that provides a mapping between IP addresses and Host names
- E.g., 155.34.50.112 => plymouth.ll.mit.edu

Internet Standards

- Internet Engineering Task Force (IETF)
  - Development on near term internet standards
  - Open body
  - Meets 3 times a year
- Request for Comments (RFCs)
  - Official internet standards
  - Available from IETF web page: http://www.ietf.org

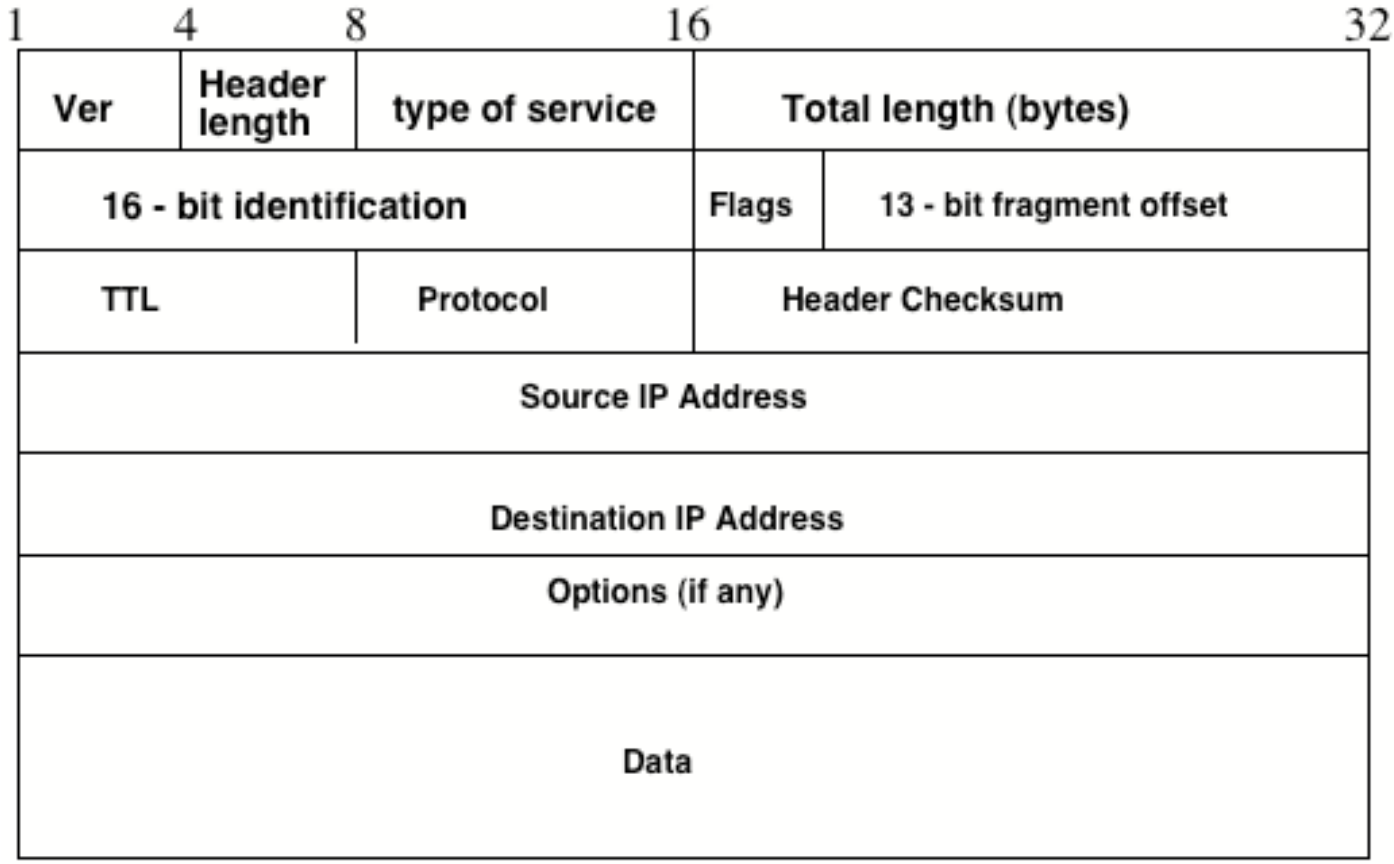
The Internet Protocol (IP)

- Routing of packet across the network
- Unreliable service
  - Best effort delivery
  - Recovery from lost packets must be done at higher layers
- Connectionless
  - Packets are delivered (routed) independently
  - Can be delivered out of order
  - Re-sequencing must be done at higher layers

- Current version V4

- Future V6
  - Add more addresses (40 byte header!)
  - Ability to provide QoS

Header Fields in IP

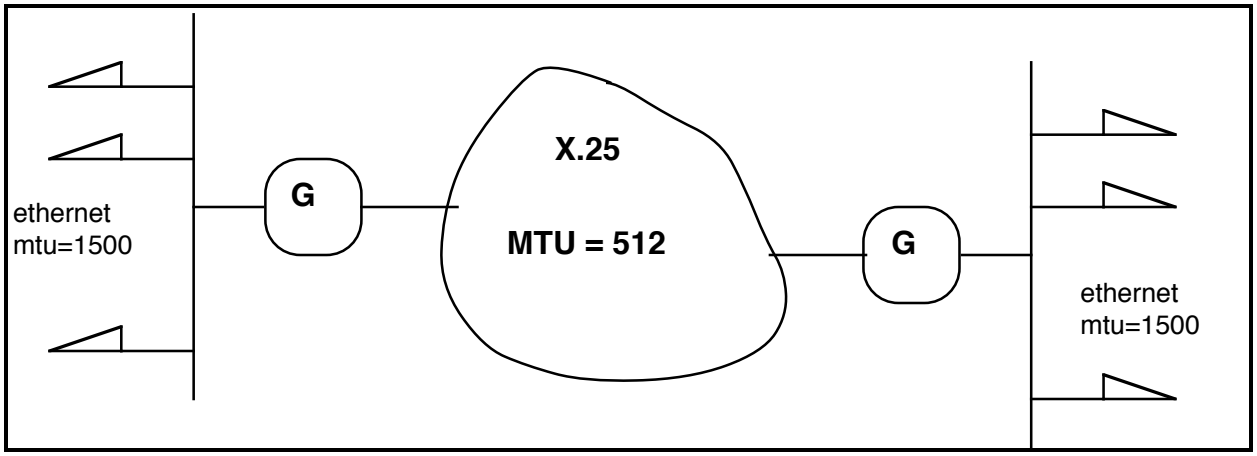


Note that the minimum size header is 20 bytes; TCP also has 20 byte header

IP HEADER FIELDS

- **Vers:** Version # of IP (current version is 4)
- **HL:** Header Length in 32-bit words
- **Service:** Mostly Ignored
- **Total length** Length of IP datagram
- **ID** Unique datagram ID
- **Flags:** NoFrag, More
- **FragOffset:** Fragment offset in units of 8 Octets
- **TTL:** Time to Live in "seconds" or Hops
- **Protocol:** Higher Layer Protocol ID #
- **HDR Cksum:** 16 bit 1's complement checksum (on header only!)
- **SA & DA:** Network Addresses
- **Options:** Record Route,Source Route,TimeStamp

FRAGMENTATION



- **A gateway fragments a datagram if length is too great for next network (fragmentation required because of unknown paths).**
- **Each fragment needs a unique identifier for datagram plus identifier for position within datagram**
- **In IP, the datagram ID is a 16 bit field counting datagram from given host**

POSITION OF FRAGMENT

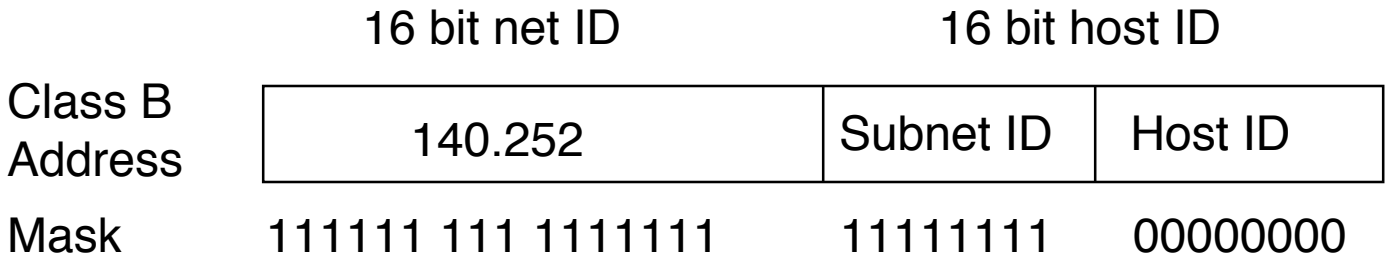
- Fragment offset field gives starting position of fragment within datagram in 8 byte increments (13 bit field)
- Length field in header gives the total length in bytes (16 bit field)
  - Maximum size of IP packet 64K bytes
- A flag bit denotes last fragment in datagram
- IP reassembles fragments at destination and throws them away if one or more is too late in arriving

IP Routing

- Routing table at each node contains for each destination the next hop router to which the packet should be sent
  - Not all destination addresses are in the routing table
    - Look for net ID of the destination "Prefix match"
    - Use default router
- Routers do not compute the complete route to the destination but only the next hop router
- IP uses distributed routing algorithms: RIP, OSPF
- In a LAN, the "host" computer sends the packet to the default router which provides a gateway to the outside world

Subnet addressing

- **Class A and B addresses allocate too many hosts to a given net**
- **Subnet addressing allows us to divide the host ID space into smaller "sub networks"**
  - **Simplify routing within an organization**
  - **Smaller routing tables**
  - **Potentially allows the allocation of the same class B address to more than one organization**
- **32 bit Subnet "Mask" is used to divide the host ID field into subnets**
  - **"1" denotes a network address field**
  - **"0" denotes a host ID field**



Classless inter-domain routing (CIDR)

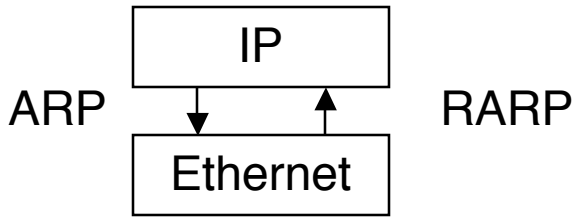
- Class A and B addresses allocate too many hosts to an organization while class C addresses don't allocate enough
  - This leads to inefficient assignment of address space
- Classless routing allows the allocation of addresses outside of class boundaries (within the class C pool of addresses)
  - Allocate a block of contiguous addresses
    - E.g., 192.4.16.1 - 192.4.32.155
    - Bundles 16 class C addresses
    - The first 20 bits of the address field are the same and are essentially the network ID
  - Network numbers must now be described using their length and value (I.e., length of network prefix)
  - Routing table lookup using longest prefix match
- Notice similarity to subnetting - "supernetting"

Dynamic Host Configuration (DHCP)

- Automated method for assigning network numbers
  - IP addresses, default routers
- Computers contact DHCP server at Boot-up time
- Server assigns IP address
- Allows sharing of address space
  - More efficient use of address space
  - Adds scalability
- Addresses are “least” for some time
  - Not permanently assigned

Address Resolution Protocol

- IP addresses only make sense within IP suite
- Local area networks, such as Ethernet, have their own addressing scheme
  - To talk to a node on LAN one must have its physical address (physical interface cards don't recognize their IP addresses)
- ARP provides a mapping between IP addresses and LAN addresses
- RARP provides mapping from LAN addresses to IP addresses
- This is accomplished by sending a "broadcast" packet requesting the owner of the IP address to respond with their physical address
  - All nodes on the LAN recognize the broadcast message
  - The owner of the IP address responds with its physical address
- An ARP cache is maintained at each node with recent mappings

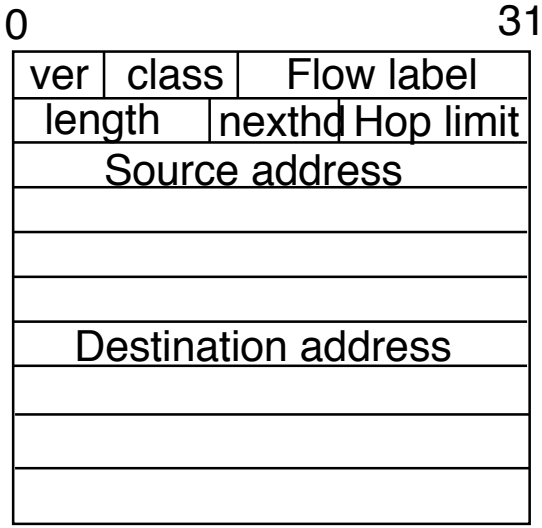


Routing in the Internet

- The internet is divided into sub-networks, each under the control of a single authority known as an Autonomous System (AS)
- Routing algorithms are divided into two categories:
  - Interior protocols (within an AS)
  - Exterior protocols (between AS's)
- Interior Protocols use shortest path algorithms (more later)
  - Distance vector protocols based on Bellman-ford algorithm
    - Nodes exchange routing tables with each other
    - E.g., Routing Information Protocol (RIP)
  - Link state protocols based on Dijkstra's algorithm
    - Nodes monitor the state of their links (e.g., delay)
    - Nodes broadcast this information to all of the network
    - E.g., Open Shortest Path First (OSPF)
- Exterior protocols route packets across AS's
  - Issues: no single cost metric, policy routing, etc..
  - Routes often are pre-computed
  - Example protocols: Exterior Gateway protocol (EGP) and Border Gateway protocol (BGP)

IPv6

- **Effort started in 1991 as IPng**
- **Motivation**
  - **Need to increase IP address space**
  - **Support for real time application - “QoS”**
  - **Security, Mobility, Auto-configuration**
- **Major changes**
  - **Increased address space (16 bytes)**
    - **1500 IP addresses per sq. ft. of earth!**
    - **Address partition similar to CIDR**
  - **Support for QoS via Flow Label field**
  - **Simplified header**
- **Most of the reasons for IPv6 have been taken care of in IPv4**
  - **Is IPv6 really needed?**
  - **Complex transition from V4 to V6**



Resource Reservation (RSVP)

- Service classes (defined by IETF)
  - Best effort
  - Guaranteed service
    - Max packet delay
  - Controlled load
    - emulate lightly loaded network via priority queueing mechanism
- Need to reserve resources at routers along the path
- RSVP mechanism
  - Packet classification
    - Associate packets with sessions (use flow field in IPv6)
  - Receiver initiated reservations to support multicast
  - "soft state" - temporary reservation that expires after 30 seconds
    - Simplify the management of connections
    - Requires refresh messages
  - Packet scheduling to guarantee service
    - Proprietary mechanisms (e.g., Weighted fair queueing)
- Scalability Issues
  - Each router needs to keep track of large number of flows that grows with the size (capacity) of the router

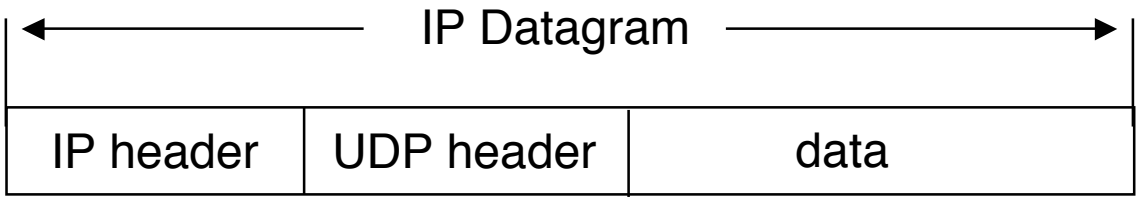
Differentiated Services (Diffserv)

- Unlike RSVP Diffserv does not need to keep track of individual flows
  - Allocate resources to a small number of classes of traffic
    - Queue packets of the same class together
  - E.g., two classes of traffic - premium and regular
    - Use one bit to differential between premium and regular packets
  - Issues
    - Who sets the premium bit?
    - How is premium service different from regular?
- IETF propose to use TOS field in IP header to identify traffic classes
  - Potentially more than just two classes

User Datagram Protocol (UDP)

- Transport layer protocol
  - Delivery of messages across network
- Datagram oriented
  - Unreliable
    - No error control mechanism
  - Connectionless
  - Not a “stream” protocol
- Max packet length 65K bytes
- UDP checksum
  - Covers header and data
  - Optional
    - Can be used by applications
- UDP allows applications to interface directly to IP with minimal additional processing or protocol overhead

UDP header format



16 bit source port number	16 bit destination port number
16 bit UDP length	16 bit checksum
Data	

- **The port numbers identifie the sending and receiving processes**
  - **I.e., FTP, email, etc..**
  - **Allow UDP to multiplex the data onto a single stream**
- **UDP length = length of packet in bytes**
  - **Minimum of 8 and maximum of 2^16 - 1 = 65,535 bytes**
- **Checksum covers header and data**
  - **Optional, UDP does not do anything with the checksum**

Transmission Control Protocol (TCP)

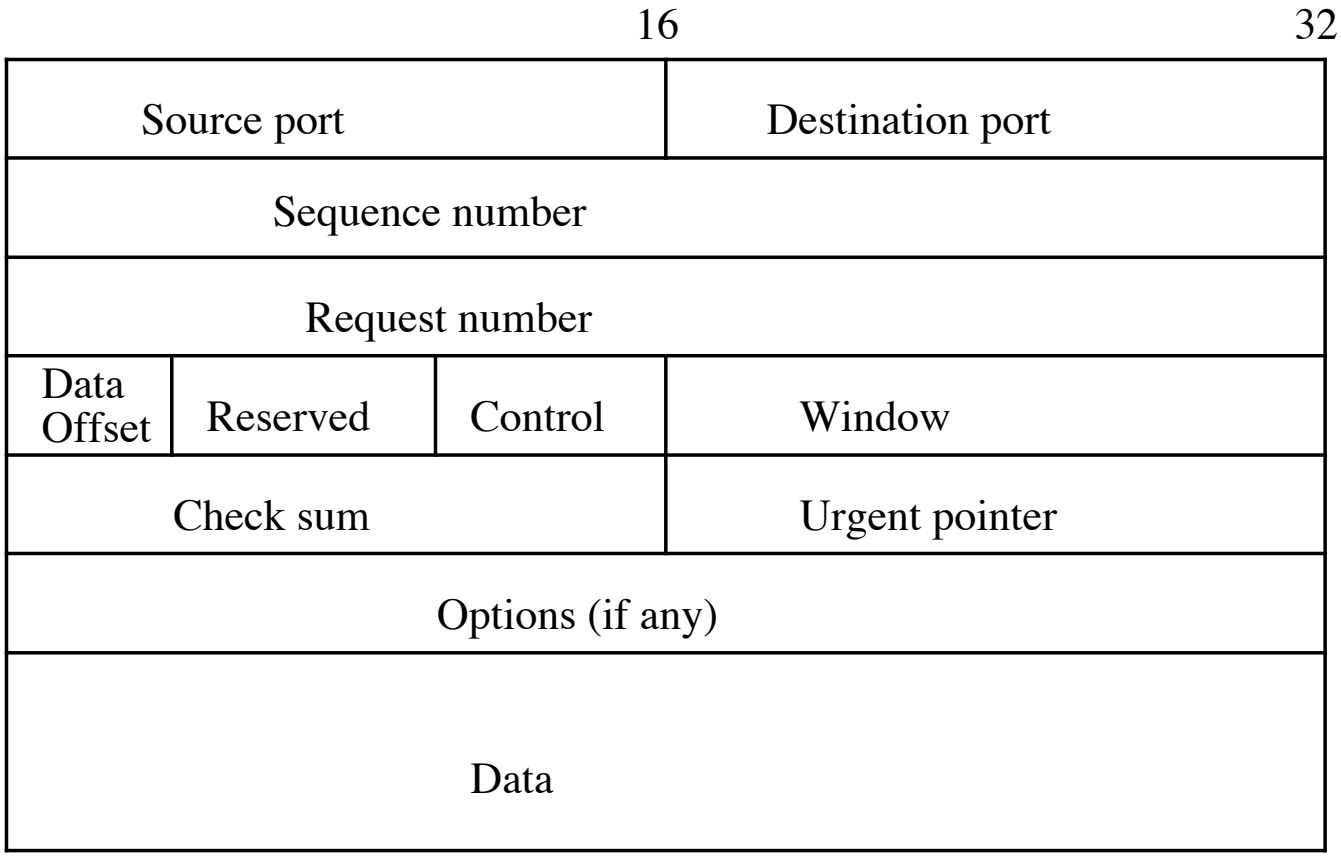
- Transport layer protocol
  - Reliable transmission of messages
- Connection oriented
  - Stream traffic
  - Must re-sequence out of order IP packets
- Reliable
  - ARQ mechanism
  - Notice that packets have a sequence number and an ack number
  - Notice that packet header has a window size (for Go Back N)
- Flow control mechanism
  - Slow start

    Limits the size of the window in response to congestion

Basic TCP operation

- At sender
  - Application data is broken into TCP segments
  - TCP uses a timer while waiting for an ACK of every packet
  - Un-ACK'd packets are retransmitted
- At receiver
  - Errors are detected using a checksum
  - Correctly received data is acknowledged
  - Segments are reassembled into their proper order
  - Duplicate segments are discarded
- Window based retransmission and flow control

TCP header fields



TCP header fields

- Ports number are the same as for UDP
- 32 bit SN uniquely identify the application data contained in the TCP segment
  - SN is in bytes!
  - It identify the first byte of data
- 32 bit RN is used for piggybacking ACK's
  - RN indicates the next byte that the received is expecting
  - Implicit ACK for all of the bytes up to that point
- Data offset is a header length in 32 bit words (minimum 20 bytes)
- Window size
  - Used for error recovery (ARQ) and as a flow control mechanism
    - Sender cannot have more than a window of packets in the network simultaneously
  - Specified in bytes
    - Window scaling used to increase the window size in high speed networks
- Checksum covers the header and data

TCP error recovery

- Error recovery is done at multiple layers
  - Link, transport, application
- Transport layer error recovery is needed because
  - Packet losses can occur at network layer
    - E.g., buffer overflow
  - Some link layers may not be reliable
- SN and RN are used for error recovery in a similar way to Go Back N at the link layer
  - Large SN needed for re-sequencing out of order packets
- TCP uses a timeout mechanism for packet retransmission
  - Timeout calculation
  - Fast retransmission

TCP timeout calculation

- **Based on round trip time measurement (RTT)**
  - **Weighted average**

$$RTT\_AVE = a*(RTT\_measured) + (1-a)*RTT\_AVE$$

- **Timeout is a multiple of RTT_AVE (usually two)**
  - **Short Timeout would lead to too many retransmissions**
  - **Long Timeout would lead to large delays and inefficiency**

- **In order to make Timeout be more tolerant of delay variations it has been proposed (Jacobson) to set the timeout value based on the standard deviation of RTT**

$$Timeout = RTT\_AVE + 4*RTT\_SD$$

- **In many TCP implementations the minimum value of Timeout is 500 ms due to the clock granularity**

Fast Retransmit

- When TCP receives a packet with a SN that is greater than the expected SN, it sends an ACK packet with a request number of the expected packet SN
  - This could be due to out-of-order delivery or packet loss
- If a packet is lost then duplicate RNs will be sent by TCP until the packet it correctly received
  - But the packet will not be retransmitted until a Timeout occurs
  - This leads to added delay and inefficiency
- Fast retransmit assumes that if 3 duplicate RNs are received by the sending module that the packet was lost
  - After 3 duplicate RNs are received the packet is retransmitted
  - After retransmission, continue to send new data
- Fast retransmit allows TCP retransmission to behave more like Selective repeat ARQ
  - Future option for selective ACKs (SACK)

TCP congestion control

- TCP uses its window size to perform end-to-end congestion control
  - More on window flow control later
- Basic idea
  - With window based ARQ the number of packets in the network cannot exceed the window size (CW)

    Last_byte_sent (SN) - last_byte_ACK'd (RN) <= CW

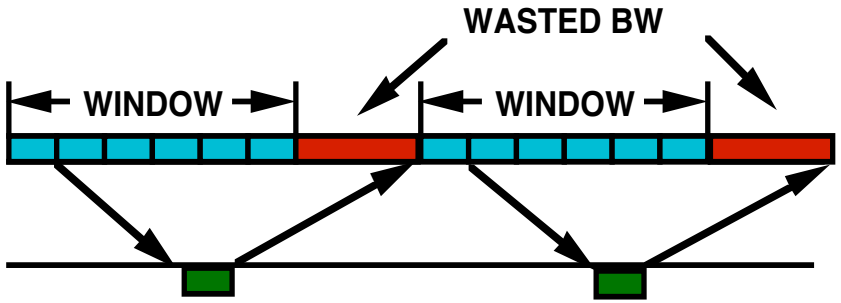
- Transmission rate when using window flow control is equal to one window of packets every round trip time

    R = CW/RTT

- By controlling the window size TCP effectively controls the rate

Effect Of Window Size

- The window size is the number of bytes that are allowed to be in transport simultaneously



- Too small a window prevents continuous transmission
- To allow continuous transmission window size must exceed round-trip delay time

Length of a bit (traveling at 2/3C)

At 300 bps	1 bit = 415 miles	3000 miles = 7 bits
At 3.3 kbps	1 bit = 38 miles	3000 miles = 79 bits
At 56 kbps	1 bit = 2 miles	3000 miles = 1.5 kbits
At 1.5 Mbps	1 bit = 438 ft.	3000 miles = 36 kbits
At 150 Mbps	1 bit = 4.4 ft.	3000 miles = 3.6 Mbits
At 1 Gbps	1 bit = 8 inches	3000 miles = 240 Mbits

Dynamic adjustment of window size

- TCP starts with CW = 1 packet and increases the window size slowly as ACK's are received
  - Slow start phase
  - Congestion avoidance phase
- Slow start phase
  - During slow start TCP increases the window by one packet for every ACK that is received
  - When CW = Threshold TCP goes to Congestion avoidance phase
  - Notice: during slow start CW doubles every round trip time

    Exponential increase!
- Congestion avoidance phase
  - During congestion avoidance TCP increases the window by one packet for every window of ACKs that it receives
  - Notice that during congestion avoidance CW increases by 1 every round trip time - Linear increase!
- TCP continues to increase CW until congestion occurs

Reaction to congestion

- Many variations: Tahoe, Reno, Vegas
- Basic idea: when congestion occurs decrease the window size
- There are two congestion indication mechanisms
  - Duplicate ACKs - could be due to temporary congestion
  - Timeout - more likely due to significant congstion

- TCP Reno - most common implementation

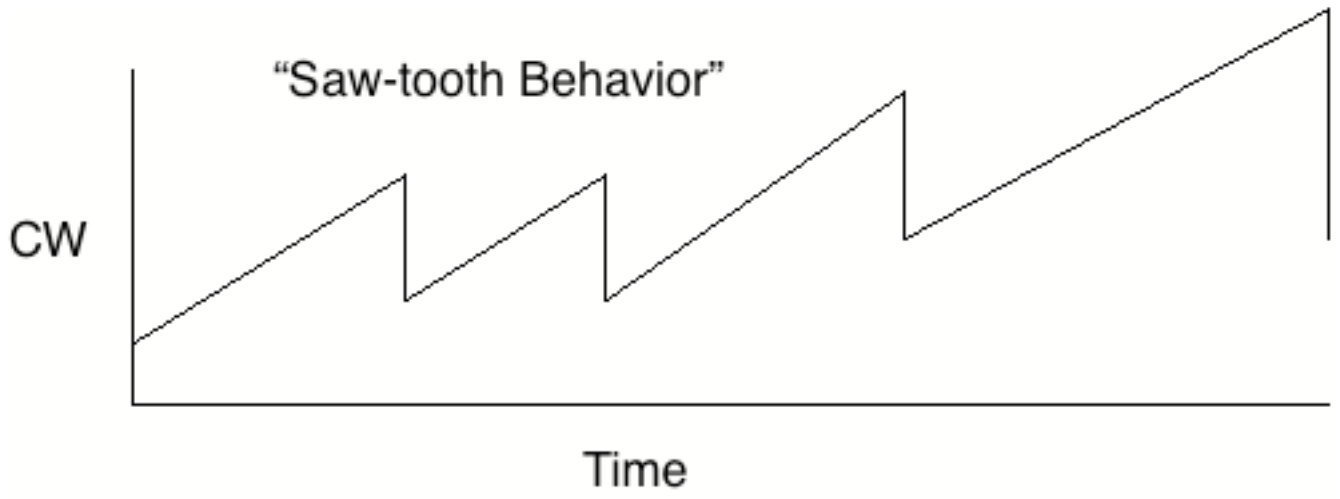
  - If Timeout occurs, CW = 1 and go back to slow start phase

  - If duplicate ACKs occur CW = CW/2 stay in congestion avoidance phase

Understanding TCP dynamics

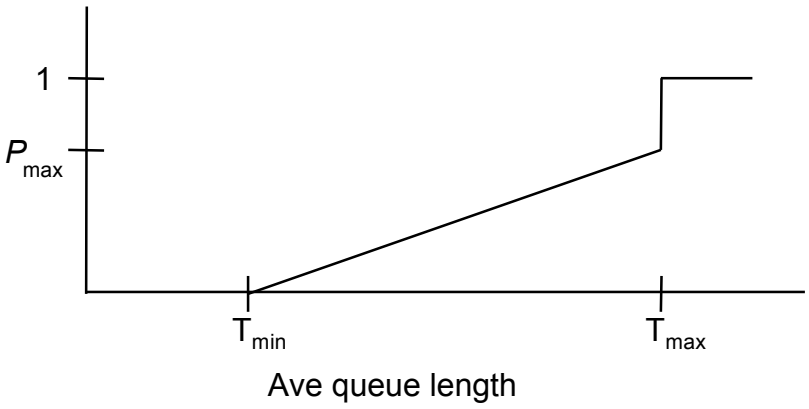
- Slow start phase is actually fast
- TCP spends most of its time in Congestion avoidance phase
- While in Congestion avoidance
  - CW increases by 1 every RTT
  - CW decreases by a factor of two with every loss

"Additive Increase / Multiplicative decrease"

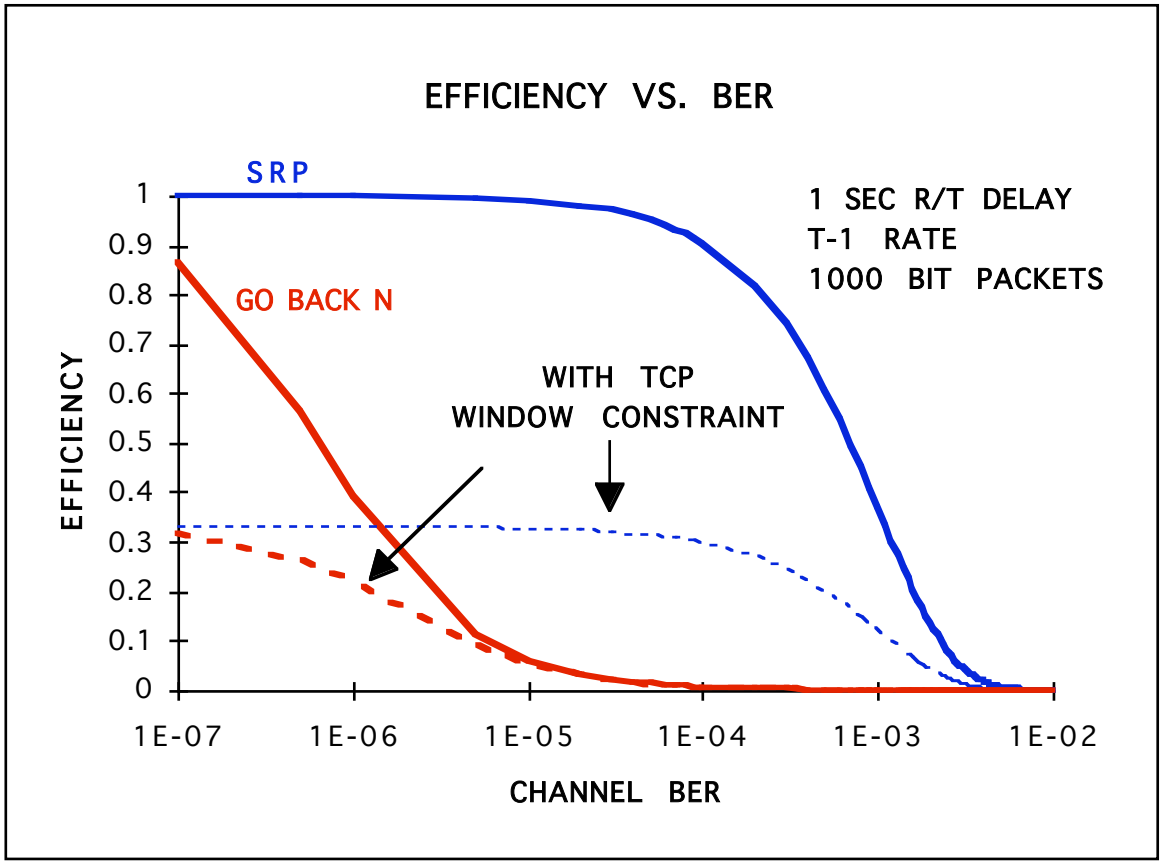


Random Early Detection (RED)

- **Instead of dropping packet on queue overflow, drop them probabilistically earlier**
- **Motivation**
  - **Dropped packets are used as a mechanism to force the source to slow down**
    - **If we wait for buffer overflow it is in fact too late and we may have to drop many packets**
    - **Leads to TCP synchronization problem where all sources slow down simultaneously**
  - **RED provides an early indication of congestion**
    - **Randomization reduces the TCP synchronization problem**
- **Mechanism**
  - **Use weighted average queue size**
    - **If AVE_Q > T_{min} drop with prob. P**
    - **If AVE_Q > T_{max} drop with prob. 1**
  - **RED can be used with explicit congestion notification rather than packet dropping**
  - **RED has a fairness property**
    - **Large flows more likely to be dropped**
  - **Threshold and drop probability values are an area of active research**

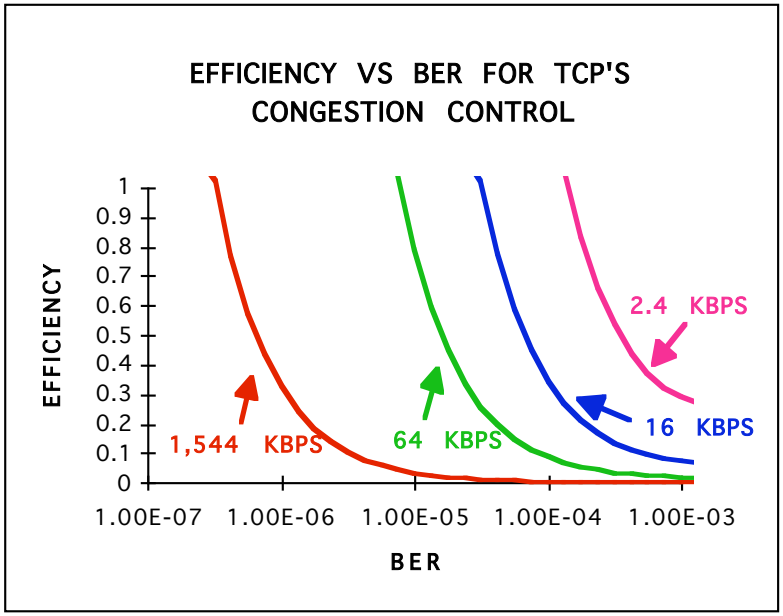


TCP Error Control



- **Original TCP designed for low BER, low delay links**
- **Future versions (RFC 1323) will allow for larger windows and selective retransmissions**

Impact of transmission errors on TCP congestion control

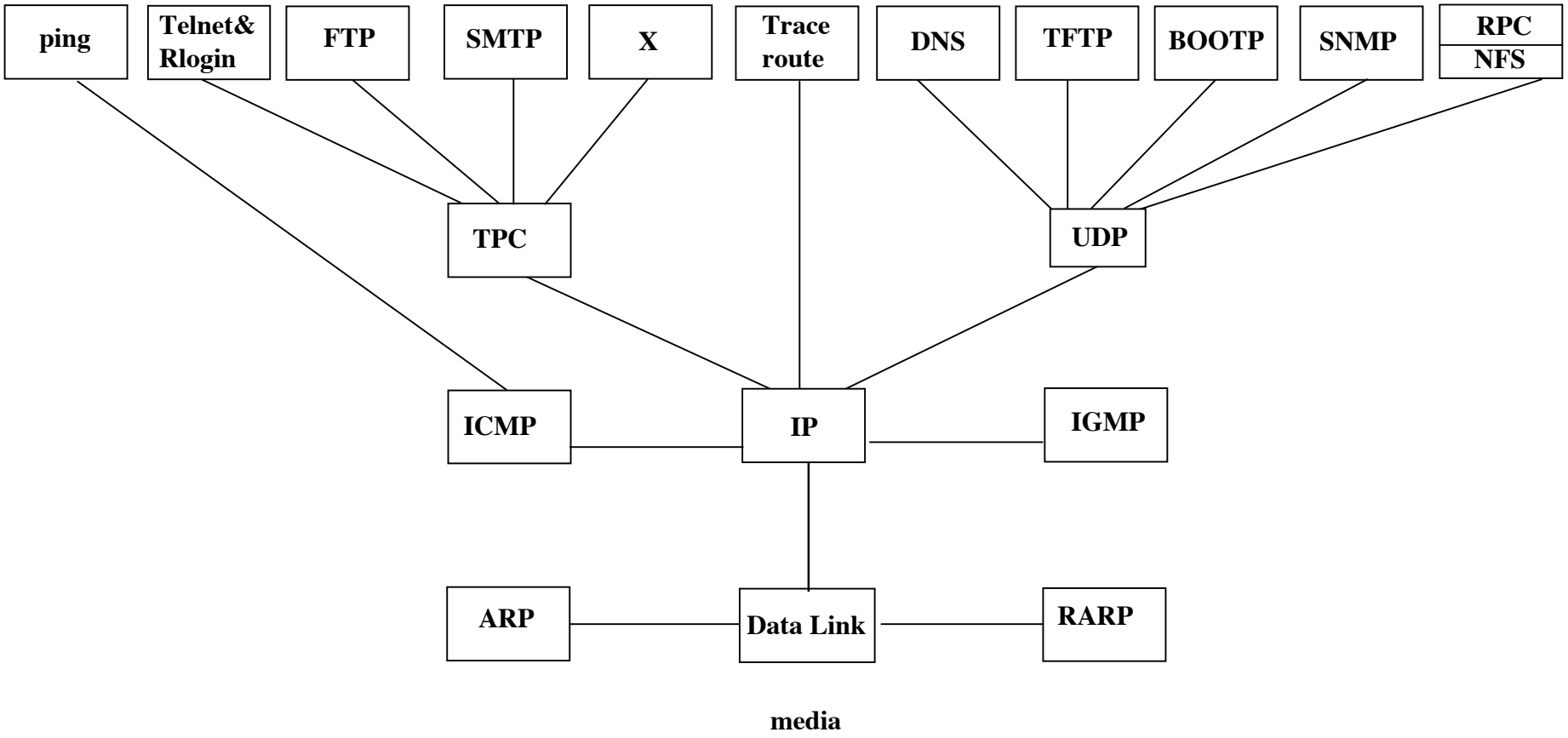


- TCP assumes dropped packets are due to congestion and responds by reducing the transmission rate
- Over a high BER link dropped packets are more likely to be due to errors than to congestion
- TCP extensions (RFC 1323)
  - Fast retransmit mechanism, fast recovery, window scaling

TCP releases

- TCP standards are published as RFC's
- TCP implementations sometimes differ from one another
  - May not implement the latest extensions, bugs, etc.
- The de facto standard implementation is BSD
  - Computer system Research group at UC-Berkeley
  - Most implementations of TCP are based on the BSD implementations
    SUN, MS, etc.
- BSD releases
  - 4.2BSD - 1983
    First widely available release
  - 4.3BSD Tahoe - 1988
    Slow start and congestion avoidance
  - 4.3BSD Reno - 1990
    Header compression
  - 4.4BSD - 1993
    Multicast support, RFC 1323 for high performance

The TCP/IP Suite

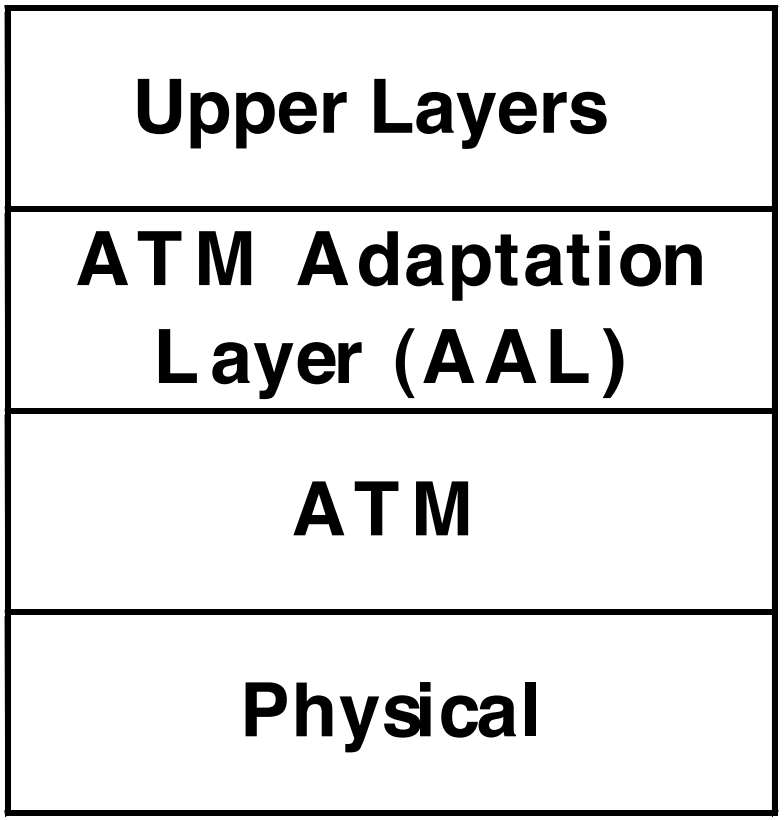


Asynchronous Transfer Mode (ATM)

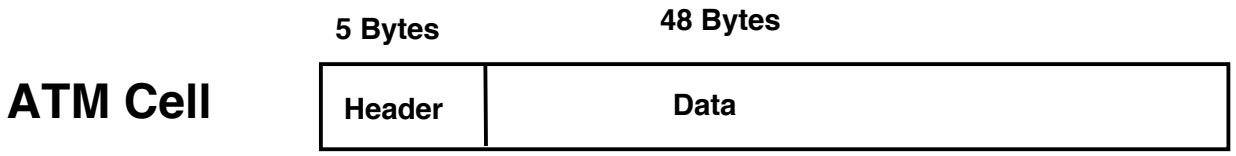
- 1980's effort by the phone companies to develop an integrated network standard (BISDN) that can support voice, data, video, etc.
- ATM uses small (53 Bytes) fixed size packets called "cells"
  - Why cells?
    - Cell switching has properties of both packet and circuit switching
    - Easier to implement high speed switches
  - Why 53 bytes?
  - Small cells are good for voice traffic (limit sampling delays)
    - For 64Kbps voice it takes 6 ms to fill a cell with data
- ATM networks are connection oriented
  - Virtual circuits

ATM Reference Architecture

- **Upper layers**
  - **Applications**
  - **TCP/IP**
- **ATM adaptation layer**
  - **Similar to transport layer**
  - **Provides interface between upper layers and ATM**
    - **Break messages into cells and reassemble**
- **ATM layer**
  - **Cell switching**
  - **Congestion control**
- **Physical layer**
  - **ATM designed for SONET**
    - **Synchronous optical network**
    - **TDMA transmission scheme with 125 μs frames**

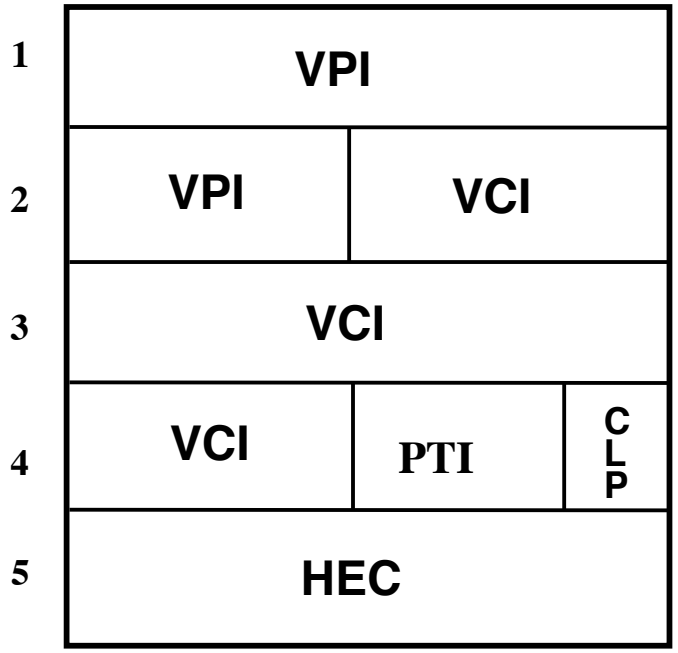


ATM Cell format



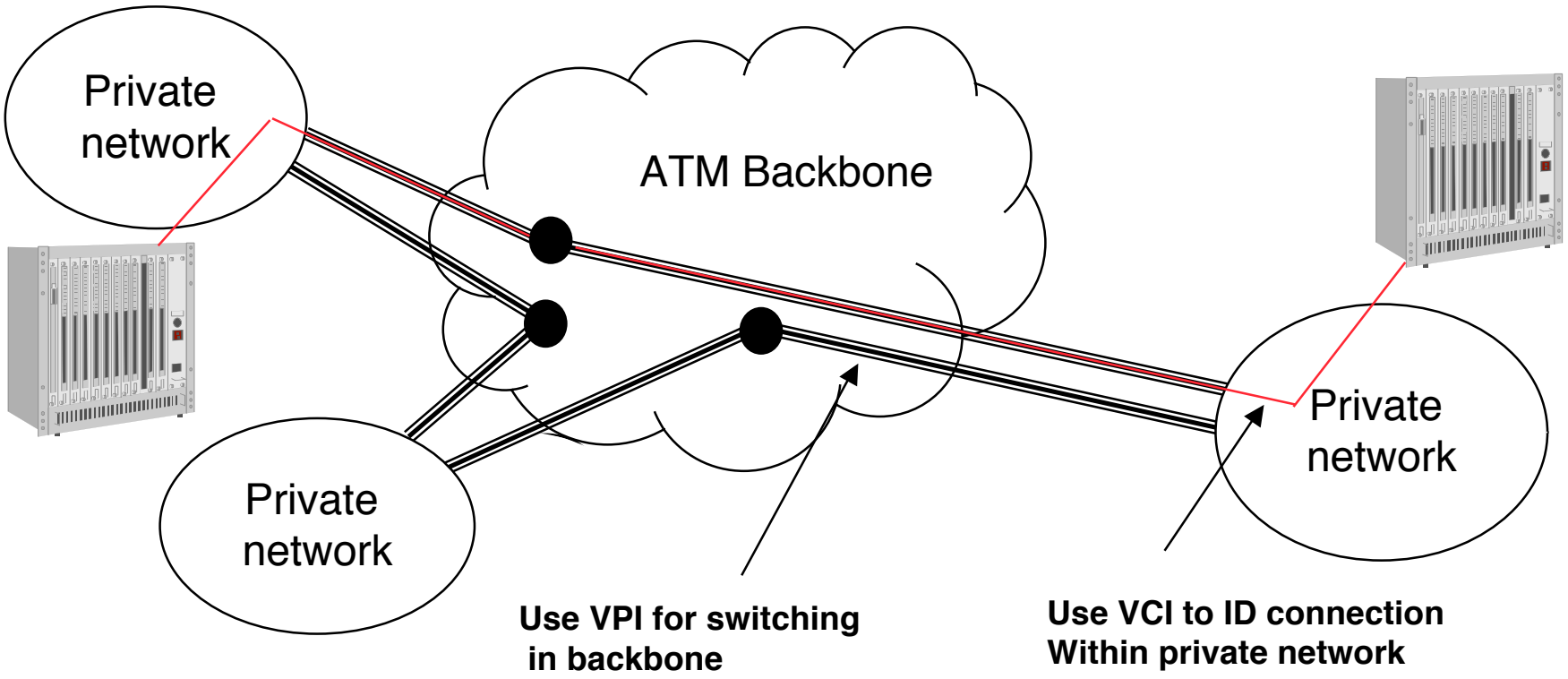
- Virtual circuit numbers (notice relatively small address space!)
  - Virtual channel ID
  - Virtual path ID
- PTI - payload type
- CLP - cell loss priority (1 bit!)
  - Mark cells that can be dropped
- HEC - CRC on header

ATM Header (NNI)



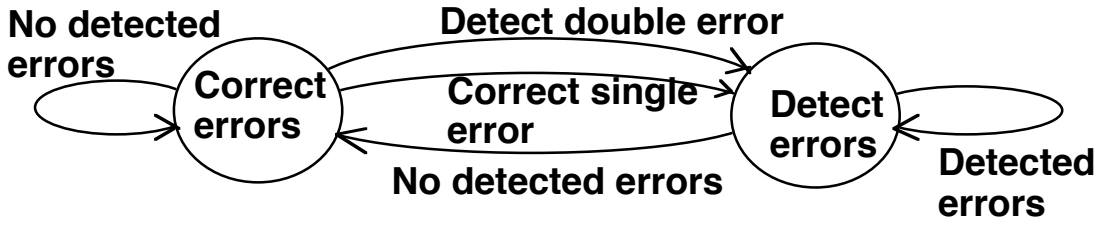
VPI/VCI

- **VPI identifies a physical path between the source and destination**
- **VCI identifies a logical connection (session) within that path**
  - **Approach allows for smaller routing tables and simplifies route computation**



ATM HEADER CRC

- ATM uses an 8 bit CRC that is able to correct 1 error
- It checks only on the header of the cell, and alternates between two modes
  - In detection mode it does not correct any errors but is able to detect more errors
  - In correction mode it can correct up to one error reliably but is less able to detect errors
- When the channel is relatively good it makes sense to be in correction mode, however when the channel is bad you want to be in detection mode to maximize the detection capability



ATM Service Categories

- **Constant Bit Rate (CBR) - e.g. uncompressed voice**
  - **Circuit emulation**
- **Variable Bit Rate (rt-VBR) - e.g. compressed video**
  - **Real-time and non-real-time**
- **Available Bit Rate (ABR) - e.g. LAN interconnect**
  - **For bursty traffic with limited BW guarantees and congestion control**
- **Unspecified Bit Rate (UBR) - e.g. Internet**
  - **ABR without BW guarantees and congestion control**

ATM service parameters (examples)

- Peak cell rate (PCR)
- Sustained cell rate (SCR)
- Maximum Burst Size (MBS)
- Minimum cell rate (MCR)
- Cell loss rate (CLR)
- Cell transmission delay (CTD)
- Cell delay variation (CDV)

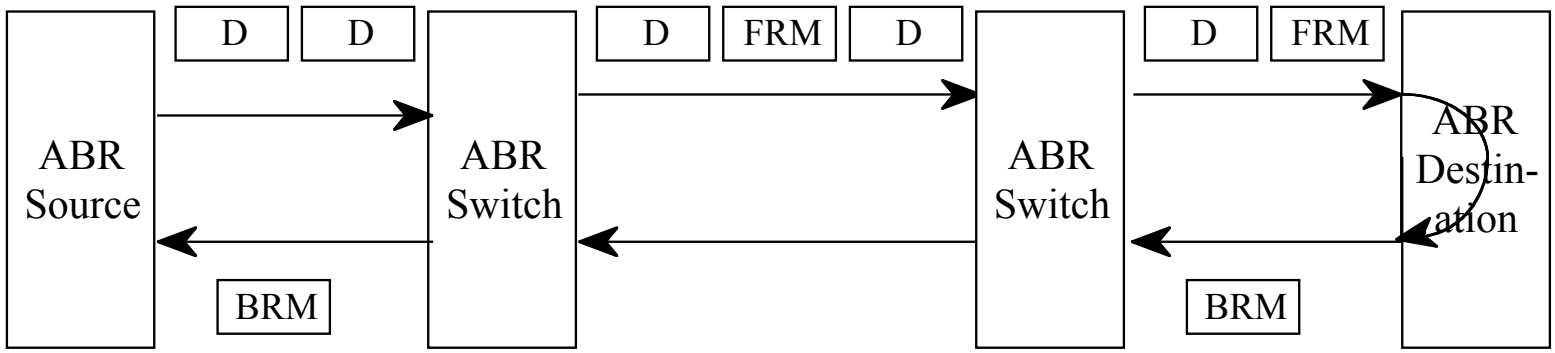
- Not all parameters apply to all service categories
  - E.g., CBR specifies PCR and CDV
  - VBR specifies MBR and SCR

- Network guarantees QoS provided that the user conforms to his contract as specified by above parameters
  - When users exceed their rate network can drop those packets
  - Cell rate can be controlled using rate control scheme (leaky bucket)

Flow control in ATM networks (ABR)

- ATM uses resource management cells to control rate parameters
  - Forward resource management (FRM)
  - Backward resource management (BRM)
- RM cells contain
  - Congestion indicator (CI)
  - No increase Indicator (NI)
  - Explicit cell rate (ER)
  - Current cell rate (CCR)
  - Min cell rate (MCR)
- Source generates RM cells regularly
  - As RM cells pass through the networked they can be marked with CI=1 to indicate congestion
  - RM cells are returned back to the source where
    - CI = 1 => decrease rate by some fraction
    - CI = 1 => Increase rate by some fraction
  - ER can be used to set explicit rate

End-to-End RM-Cell Flow



D = data cell

FRM = forward RM cell

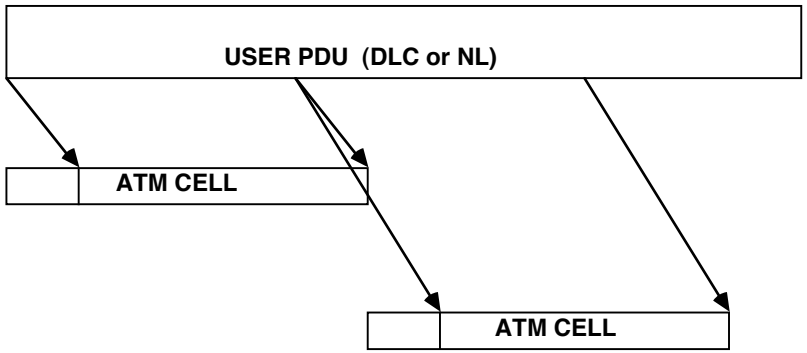
BRM = backward RM cell

At the destination the RM cell is "turned around" and sent back to the source

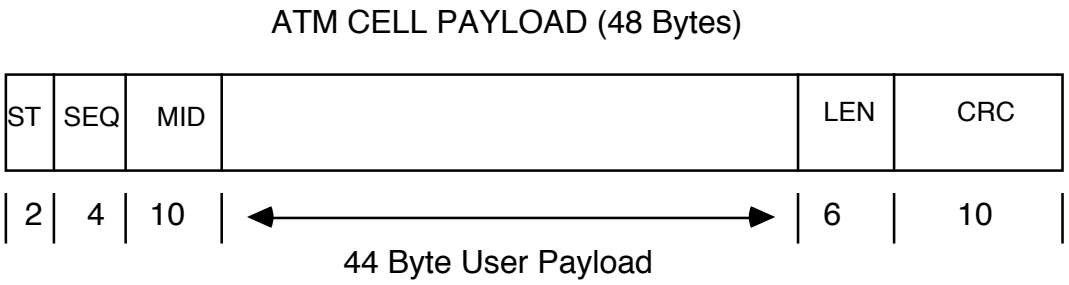
ATM Adaptation Layers

- Interface between ATM layer and higher layer packets
- Four adaptation layers that closely correspond to ATM's service classes
  - AAL-1 to support CBR traffic
  - AAL-2 to support VBR traffic
  - AAL-3/4 to support bursty data traffic
  - AAL-5 to support IP with minimal overhead
- The functions and format of the adaptation layer depend on the class of service.
  - For example, stream type traffic requires sequence numbers to identify which cells have been dropped.

Each class of service has A different header format (in addition to the 5 byte ATM header)



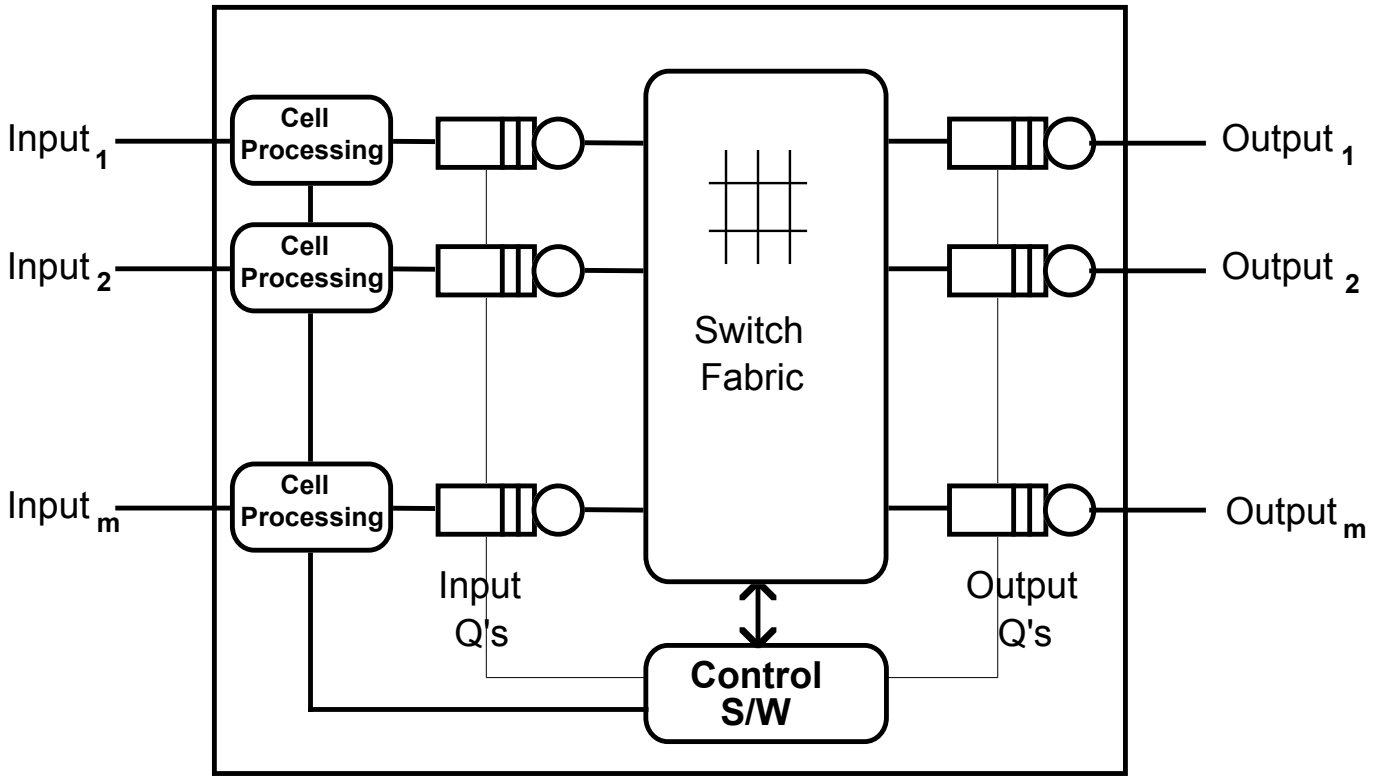
Example: AAL 3/4



- **ST: Segment Type (1st, Middle, Last)**
- **SEQ: 4-bit sequence number (detect lost cells)**
- **MID: Message ID (reassembly of multiple msgs)**
- **44 Byte user payload (~84% efficient)**
- **LEN: Length of data in this segment**
- **CRC: 10 bit segment CRC**

- **AAL 3/4 allows multiplexing, reliability, & error detection but is fairly complex to process and adds much overhead**
- **AAL 5 was introduced to support IP traffic**
  - **Fewer functions but much less overhead and complexity**

ATM cell switches



- **Design issues**
  - **Input vs. output queueing**
  - **Head of line blocking**
  - **Fabric speed**

ATM summary

- ATM is mostly used as a “core” network technology
- ATM Advantages
  - Ability to provide QoS
  - Ability to do traffic management
  - Fast cell switching using relatively short VC numbers
- ATM disadvantages
  - It not IP - most everything was design for TCP/IP
  - It’s not naturally an end-to-end protocol
    - Does not work well in heterogeneous environment
    - Was not design to inter-operate with other protocols
    - Not a good match for certain physical media (e.g., wireless)
  - Many of the benefits of ATM can be “borrowed” by IP
    - Cell switching core routers
    - Label switching mechanisms

Multi-Protocol Label Switching (MPLS)

“As more services with fixed throughput and delay requirements become more common, IP will need virtual circuits (although it will probably call them something else)”

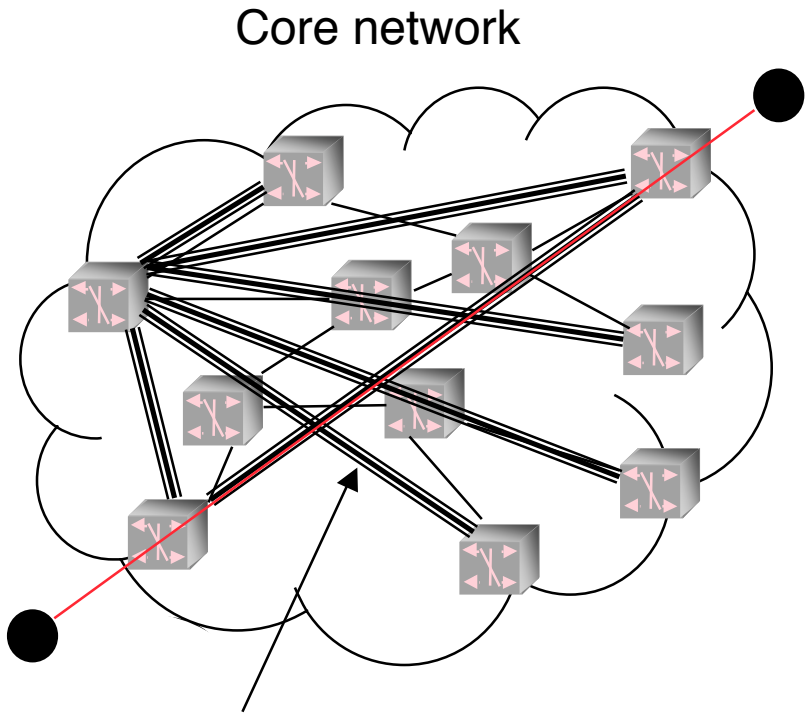
Data Networks lecture notes - April 28, 1994

Label Switching

- Router makers realize that in order to increase the speed and capacity they need to adopt a mechanism similar to ATM
  - Switch based on a simple tag not requiring complex routing table look-ups
  - Use virtual circuits to manage the traffic (QoS)
  - Use cell switching at the core of the router
- First attempt: IP-switching
  - Routers attempt to identify flows
    - Define a flow based on observing a number of packets between a given source and destination (e.g., 5 packets within a second)
  - Map IP source-destination pairs to ATM VC's
    - Distributed algorithm where each router makes its own decision
- Multi-protocol label switching (MPLS)
  - Also known as Tag switching
  - Does not depend on ATM
  - Add a tag to each packet to serve as a VC number
    - Tags can be assigned permanently to certain paths

Label switching can be used to create a virtual mesh with the core network

- Routers at the edge of the core network can be connected to each other using labels
- Packets arriving at an edge router can be tagged with the label to the destination edge router
  - "Tunneling"
  - Significantly simplifies routing in the core
  - Interior routers need not remember all IP prefixes of outside world
  - Allows for traffic engineering
    - Assign capacity to labels based on demand



References

- TCP/IP Illustrated (Vols. 1&2), Stevens
- Computer Networks, Peterson and Davie
- High performance communication networks, Walrand and Varaiya