Random Variable

Sampel space. S, set of all outcomes
Random Variable. X assigns a number to every element $s \in S$
Expectation. Represents the long-run average value of repetitions of the experiment. Finite $E[X] = x_1 Pr(X = x_1) + ...x_k Pr(X = x_k)$. Countably infinite $E[X] = \sum_{i=1}^{\infty} x_i Pr(X = x_i)$. Absolutely continuous $E[X] = \int_{-\infty}^{\infty} xf(x)dx$ where $f(x)$ is the probability density function.

Probability

Conditional Probability. $Pr(A \mid B)$, A given that B occurred. $Pr(A \mid B) = \frac{Pr(A \cap B)}{Pr(B)}$, $Pr(B \mid A) = \frac{Pr(A \cap B)}{Pr(A)}$
Properties. $0 \leq Pr(A \mid B) \leq 1$. $Pr(S \mid B) = 1$. If $B_1, B_2...$ are mutually exclusive, then $Pr(\bigcup_{i=1}^{\infty} B_i|A) = \sum_{i=1}^{\infty} Pr(B_i|A)$. If B_1 and B_2 are disjoint, then $Pr(B_1 \cup B_2|A) = Pr(B_1|A) + Pr(B_2|A)$
Total Probability. Mutually exclusive $(A_i \cap A_j = \emptyset)$ and exhaustive $(\bigcup_{i=1}^{n} A_i = S)$, then for any event B, $Pr(B) = \sum_{i=1}^{n} Pr(B \cap A_i) = \sum_{i=1}^{n} Pr(A_i)Pr(B|A_i)$

Let X and Y be two independent exponential random variables with parameters λ_x and λ_y. What is the probability $P\{X > Y\}$?

$$P\{X > Y\} = \int_0^\infty P\{X > Y | Y = y\} f_Y(y)\, dy$$
$$= \int_0^\infty P\{X > y\}(\lambda_y e^{-\lambda_y y})\, dy = \int_0^\infty e^{-\lambda_x y}(\lambda_y e^{-\lambda_y y})\, dy$$
$$= \lambda_y \int_0^\infty e^{-(\lambda_x + \lambda_y)y}\, dy$$
$$= \lambda_y \left(-\frac{1}{\lambda_x + \lambda_y} e^{-(\lambda_x + \lambda_y)y} \Big|_0^\infty \right) = \frac{\lambda_y}{\lambda_x + \lambda_y}$$

Bayes Theorem. Let $A_1, A_2, ...A_n$ be a partition of sample space S. Then $Pr(A_k|B) = \frac{Pr(A_k)Pr(B|A_k)}{\sum_{i=1}^{n} Pr(A_i)Pr(B|A_i)}$, where $Pr(B)$ is just the denominator.
Independent Events. $Pr(A \cap B) = Pr(A)Pr(B)$. Independent events refer to events that don't influence other events. **Mutually independent implies pairwise independence, but not the other way**. If there is a function $f_{X,Y}(x, y)$ that is independent, it can be expressed as $f_{X,Y}(x, y) = f_X(x)f_Y(y)$
Distribution Function. $F(x)$. Probability that X takes on some rang eof values, y is the same as the sum of applying X to all values from $-\infty$ to y.
Probability Density Function. X is continuous if there is a function $f(x)$ that is a derivative of the distribution function. $f(x) = \frac{d}{dx}F(x)$
Expectation. $\int_{-\infty}^{\infty} xf(x)dx$ if given the pdf or $\sum_{x=-\infty}^{\infty} xPr(X = x)$ if given just random variable.
Exponential Distribution. Events happen one at a time. Time is measured for how long it takes for some event to happen. Waiting time gets shorter means that the probability of event happening over longer periods of time decrease exponentially. Basically the distribution of time between events. Average number of events per unit time. Similar to Little Law. Geometric distribution is also memoryless; only these 2 are memoryless
Formula. If $\lambda \geq 0$, for $x \geq 0$, $F(x) = Pr(T \leq x) = 1 - e^{-\lambda x}$ and $f(x) = \frac{dF(x)}{dx} = \lambda e^{-\lambda x}$.
Mean. $\int_{-\infty}^{\infty} xf(x)dx = \frac{1}{\lambda}$
Unit. $\frac{1}{\text{unit time}}$
Memoryless Property. No biasness and every time is like the first time. The value of s does not affect the final result.
Formula. $Pr(T > s+t|T > s) = Pr(T > t), \forall s, t \geq 0$

$$P\{T > s + t \mid T > s\} = \frac{P\{T > s + t, T > s\}}{P\{T > s\}}$$
$$= \frac{P\{T > s + t\}}{P\{T > s\}}$$
$$= \frac{1 - P\{T \leq s + t\}}{1 - P\{T \leq s\}}$$
$$= \frac{1 - (1 - e^{-\lambda(s+t)})}{1 - (1 - e^{-\lambda s})} = e^{-\lambda t}$$

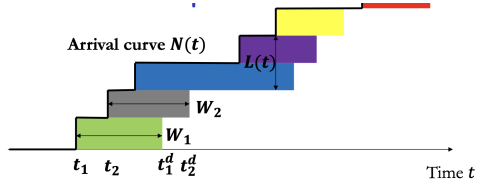
Deriving minimum probability.
$Pr(min(X_1, X_2) > t) = Pr(X_1 > t, X_2 > t)$. Because they are independent, we can multiply their probabilities together. What this means is that we can conver $min(X_1, X_2)$ to be T' and perform Poisson calculations on T' instead.
$Pr(T_2 > T_1) = \frac{\lambda_1}{\lambda_1 + \lambda_2}$

Metrics

Link Rate/Bandwidth/Capacity. All refers to the same thing; the number of bits being **pushed** per second. Think of of link. Packets are things to be put on the truck. Transmission delay is the time to put 1 packet on the truck.
Throughput. Of a flow, refers to the number of bits communicated per unit time. Transferred from **end to end**. How many things can we put on the truck?
End-to-end Delay. Processing + queueueing + transmission.
Store-and-forward. Receive the entire packet before making the hop. If transmitted wirelessly, then the transmission speed would be the same as the travelling speed through the medium. The end-to-end delay would be the **sum** of transmission.
Bandwidth. Affects transmission delay. If the amount of things we can put onto the link is very little, the bandwidth would be very little.
Packet size. Affects transmission and processing. If the size of the packet to be placed on the link is very big, the transmission delay would be huge. Similarly for processing the packet.
Distance. Affects propogation delay. If the distance is very long, it would take more time to go from one end to the other.
Response Time. Time from pressing enter and getting something on the browser. It is normally 1 round trip time which is **2x** the end-to-end delay since it goes from client to server and back.
Statistical Multiplexing. Bandwidth being shared on demand, no fixed timing pattern.
Circuit switching. Depending on the capacity and requirements, we allocate equally.
Packet switching. Sending of packets over different mediums and ways as long as they all end up at the same place.
Queueing Delay. In the case of more inputs than outputs, its possible to accommodate. Increase chances of more active users at the same time. But the service would not be guarenteed. Optimise for number of users.
Perks of packet switching. Great for asynchronous bursty data since it allows resource sharing and there is no call setup. Excessive congestion - possibly leading to packet delay and loss. Protocols needed for reliable data transfer and congestion control. Server that serves the packet in the queue, based on link capacity.
Probability that at least m users are online.

- $Pr(X > m) = Pr(X = m + 1)...Pr(X = N)$
- Calculate $Pr(X = n)$ as a product of the number of people online and offline.
- $Pr(X = n) = \binom{N}{n} p^n (1 - p)^{N-n}$
- $Pr(X > m) = \sum_{n=m+1}^{N} \binom{N}{n} p^n (1 - p)^{N-n}$

Little Law. How many people on average are there in the line at any time. $L = \lambda W$



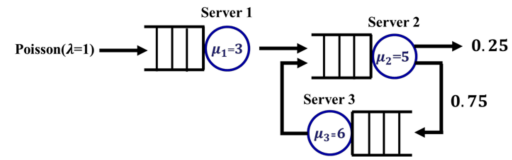
- t_i and t_i^d : arrival and departure time of ith customer
- $N(t)$: # of customers arrived till time t
- $L(t)$: # of customers in the system at time t
- W_i: sojourn time of customer
  - Fact: $t_i^d = t_i + W_i$

Formulas.

- $\lambda = \lim_{t \to \infty} \frac{N(t)}{t}$
- $W = \lim_{t \to \infty} \frac{1}{n} \sum_{j=1}^{n} W_j$
- $L = lim_{t \to \infty} \frac{1}{t} \int_0^t L(s)ds$

Queueing

Inter-arrival Time. Time before the next packet arrives is $T_i = t_{i+1} - t_i$. Model arrival time by independent and identically distributed random variable. We don't care about other T_i's. Identically distributed means that they follow the same schedule as other arrival times.
Poisson Process. Process that carries out some event one at a time. In no set schedule and based on a n average rate. The combination of Poisson processes is still a Poisson process.
Packet properties. Starting time of the process does not matter because of the memoryless property. Merging 2 processes together just adds their rate together.
Service Time. Average service time would be $E[S] = \frac{1}{\mu}$
MM1 model. Single server with infinite queue size. Poisson arrival rate of λ. Exponential service time with rate of μ. Arrival and service time are independent. FIFO service. Relationship between Q and L is $E[L] = E[Q] + E[\#\text{ of users being server}] = E[Q] + \rho$. For any 2 flows that can be supported by μ_1, μ_2, when merging them both and serving them by the aggregated capacity μ, $E[W] < min(E[W_1], E[W_2])$, similarly for queueing delay.
Utilisation. Percentage of time that server is busy or the probability that random observation finds server busy. Not limited to MM1. $\rho = \frac{\lambda}{\mu}$. System would be stable if $\lambda < \mu$.
Stability. Room for variability (capacity to handle varying workload), buffer against overloads, maintaing QoS, prevent queues and delays (faster processing time), scheduled maintainence and upgrades.
Other Formulas. $\rho = \frac{W}{W+1} = \frac{L}{\lambda(L+1)}$
Birth-Death Process. Going from between states is simply enqueueing and dequeueing.
Packets in system. π_i is the percentage of time that exactly i packets are in the **system (server + queue)** or the probability that a random observation finds i packets in the system. $\pi_i = Pr(L = i) = \rho^i(1 - \rho)$. The sum of all π_i is 1. And it follows a geometric distribution.
MM1 Mean Sojourn times. Empty queue but server busy (service time of packet + service time of packet being served) : $\frac{2}{\mu}$.
Geometric Distribution. Density function is $Pr(L = i) = \rho(1 - \rho)^i$.
Formulas. Average number of packets in **system**: $E[L] = \frac{\rho}{1-\rho}$. Average time packets in **system**: $E[W] = \frac{1}{\mu-\rho}$. Average number of packets in **queue**: $E[Q] = \frac{\rho^2}{1-\rho}$. Average queueing delay in **queue**: $E[D] = E[W] - \frac{1}{\rho}$. For $i \geq 1$, $Pr(Q = i) = Pr(L = i + 1) = \pi_{i+1}$. When $i = 0$, $Pr(Q = 0) = Pr(L = 0) + Pr(L = 1) = (1 - \rho) + \rho(1 - \rho)$. For the system, we can sum of the values of λ. **Serving flows separately means they are each their own system now**. For individual packets, calculate the total average number of packets then apply little law.
Effective bandwidth. Physical link capacity. Refers to the actual throughput that can be achieved, depending on the QoS achieved. Physical capacity of link is the maximum achievable throughput if utilisation is below some ρ and queueing delay is bounded by some value.
Statistical Multiplexing vs TDM. Allocating each Poisson stream its own queue results in the same utilisation Substitute ρ, μ by $\frac{\rho}{k}, \frac{\mu}{k}$ where k is the number of streams.
Formulas. $E[W] = \frac{1}{\rho}\frac{1}{1-\rho}$ Means that the time the packets stay in the system increase by k time. $E[Q] = \frac{\rho^2}{1-\rho}$, $E[L] = \frac{\rho}{1-\rho}$. Average packet waiting and total users also k times larger.
Burke's Theorem. If MM1 with arrival rate λ starts in a steady state. The **departure process** is Poisson with rate λ. Number of customers in system at any time t is independent of the departure time prior to time t. Utilisation of μ_i is $\rho_i = \frac{\lambda}{\mu_i}$. Calculate the joint probability by multiplying them together because of independence.
Jackson Network.



To determine $E[W]$, obtain the utilisation of each

server, then apply formula to get $E[W]$. Calculate the effective arrival from external arrival and feedback arrivals from other systems. $\lambda_i = r_i + \sum_{j=1}^{n} \lambda_j P_{ji}$. For step 2, we can calculate average number of packets in the whole system and apply little law, where λ is the total rate of packets entering the system. $E[L_i] = \frac{\rho_i}{1-\rho_i}$. Sum up all the $E[L_i]$ and apply little law. Or we sum up the average $E[W]$ in each subsystem. $E[W] = \sum_{i=0}^{n} E[V_i]E[W_i]$. V_i is the number of loops a packet spends in a subsystem. **Stability.** Look at each subsystem to determine the μ_i that it should have for stability. **Minimising average quantity.** . Perform differentiation. Since the result should be some constant value, differentiating the minimum quantity would lead to a result of 0. Then solve equations.

$$minimize\ \frac{1}{\mu_1 - \lambda} + \frac{4}{\mu_2 - 4\lambda};\ subject\ to: \begin{cases} \mu_1 > \lambda \\ \mu_2 > 4\lambda \\ \mu_1 + \mu_2 = \mu \end{cases}$$

$$\Rightarrow \begin{cases} minimize\ f(x) = \frac{1}{x - \lambda} + \frac{4}{\mu - x - 4\lambda} \\ subject\ to: \lambda < x < \mu - 4\lambda \end{cases}$$

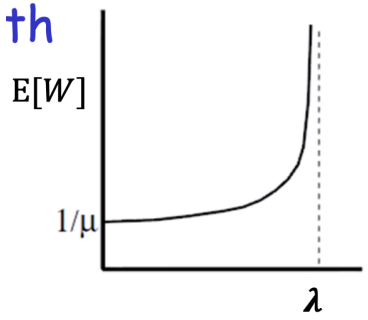
$$f'(x) = -(x-\lambda)^{-2} + 4(\mu - x - 4\lambda)^{-2}$$

$$f'(x^*) = 0 \Rightarrow (2(\mu - x^* - 4\lambda)^{-1})^2 = \left((x^* - \lambda)^{-1}\right)^2$$
$$\Rightarrow 2(\mu - x^* - 4\lambda)^{-1} = (x^* - \lambda)^{-1}$$
$$\Rightarrow 2(x^* - \lambda) = \mu - x^* - 4\lambda \Rightarrow 3x^* = \mu - 2\lambda$$

$$\Rightarrow \begin{cases} \mu_1^* = \frac{1}{3}\mu - \frac{2}{3}\lambda \\ \mu_2^* = \frac{2}{3}\mu + \frac{2}{3}\lambda \end{cases}$$

Resource allocation

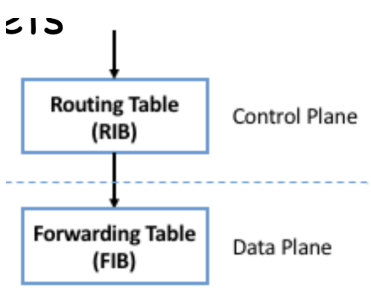
Effective bandwidth. Geometric distribution of average waiting time.



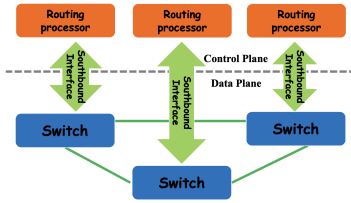
Definition. More capacity allows higher throughput and provides lower delay.
Delay-throughput correlation. More throughput means lower delay, means that the inverse of delay increases.
Achieved throughput. $\lambda = \mu - \frac{1}{E[W]}$
Resource allcoation. 2 packet flows going through a link, each with different rates. Flows are λ_i and link has a capacity of μ.
Methodology. Split capacity into μ_i and serve the flows differently. Since flows are different, even if we split the capacity equally, the delays would be different. To guarentee delays, the sum of the flows should be bounded.
Reality. There may not be guarentee for packet delivery. Also, real time apps would suffer performance issues. Instead, allocate different amount of resources to different app flows. Use user happiness as a metric for performance.
Fairness. **Equal** sharing of resources. Optimise for those who want less - give them all they want. For larger demands, split them equally. To determine who is less, comapre the difference between each and group them base on the difference.
Max-min Fairness solution. For each resource, split equally for each link. If any of the flows exceeds their demands, truncate to their demands. Split the remainder amongst the other links for that resource. Repeat until no demands are exceeded. For the flow that was fulfilled, we ignore the capacities that the satisfied flow passes through only if there is only 1 other flow passing through that same capacity. For the flow that appears the most times, we take the min of all the values. When 2 flows remain, just take the min of each one. Everytime we solve a flow, split (if needed) the extra resources amongst the other flows that have not been solved.
Uniqueness. There is always a unique min-max solution.
Bottleneck Resource. Resource is satuated and flow has maximum amount of data that the link can carry is already being transmitted (compare the normalised allocation, the one including weights). Any additional data trying to go throught the link will be delayed or dropped since the link can't handle it. Flow has the maximum rate among all flows using resource. Flow cannot get more resource if allocation is fair; otherwise it will hurt flows with lower rate; the current resource is already distributed fairly. To calculate, determine which resource is satuated (ie fully utilised), then take the max flow (x_i) to tell us which resource got bottlenecked.
Water filling algorithm. List the demands as empty buckets. Smaller demands can be easily fulfilled so we optimise for that. Ensure that everyone gets at least the same amount.
Weighted Max-Min. To calculate the share at each resource, $x_i = \frac{\phi_i}{\sum^j \phi_j} C$, where ϕ_i is the weight of x_i, C is the capacity of that resource, $\sum^j \phi_j$ is the sum of all the weights of links. Similar to unweighted, truncate and distribute **according to weights**.
Bottleneck. Same condition except now to determine flow, we need to normalise it ($\frac{x_i}{\phi_i}$).

SDN - Design Principle

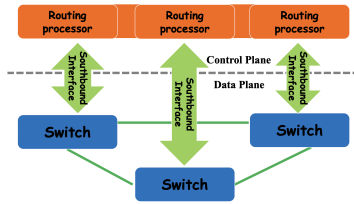
Data Plane. Process or deliver packets based on the state in routers and endpoints. uses IP, TCP, Ethernet, etc. Fast timescales per packet. Handles packet itself.
Control Plane. Establish router state. Determines how/where packets are forwarded. Routing traffic engineering, firewall. Slow timescales per control even. Handles how to deliver the packets.



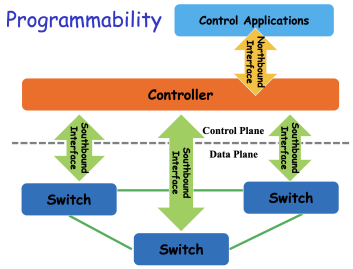
Disaggregation. Should be codified in an open interface. Have an interface that allows communication between process and switch. Mainly communication from processor to switch.



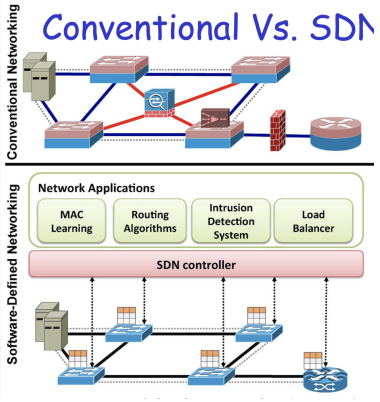
Implications. Allows network opertors to purchase plans from different vendors as long as the same interface is provided. Data planes consist of cheaper hardware devices. Forwarding abstraction needs to defined; a way for control plane to tell data plane to forward packets in a particular way.
Flow Rule. Mach action pair and any packet that was matched have an associated action applied to it. Looks at packet header.
Flow Table. Multiple in pipeline. Perform logic at different stages. Specifies what to do with the packet, more general than matching to some port.
Benefits. New market landscape. Software control of network can evolve independent of hardware. Allows horizontal scaling.
Control. Making real-time decisions about how to respond to link and switch failures. Control planes must learn about failure and provide remedy within milliseconds.
Configuration. Operators need to configure switch and routers to update RIB. Interface is capable of installing new routes which seems like installing new flow rule.
Control Plan Implementation. Run software that implements control plane on switch. Switch operates autonomously and communicate with peer switches throughout network to construct routing tables. Off-switch implementation - control plan is independent of data plane and logically centralised. Running controller away from swithces. On/inside switch is an independent operation, can talk to other planes. Outside switch is logically centralised.
Centralised Control.



Processors in control plane are combined into 1 big processor. Need to know how to handle corner case with other pages. May consist of multiple controllers but requires talking to each other to give centralised view.
Benefits. Decisions are easier to make. Logically vs physically centralised.
Data plane implementation. FIB to forwarding pipeline. Each table focus on a subset of header fields that might be involved in a given flow rule. Packet is processed by multiple tables in sequence to determine how its forwarded.
Fixed function data plane. Uses header fields (easy to compute offsets in every packet and are well-known). Assumes packet header format.
Programmable data plane. Performance optimisations, potential changes in protocol.

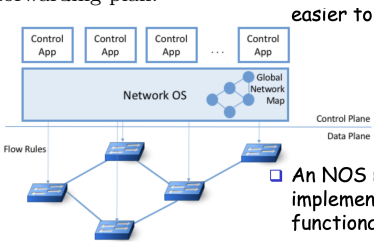


Programmability. Routing table where each table focus on an individual task. Chain of responsibility of tables. Write control app through some northbound API.
Benefits. Easy network management. Expressed as policies and easier to debug and verify behavior. Rapid innovation and fast evolution. Enable new services and better performance. Detailed configuration handled by controller. Control shift from vendors to operators to users.

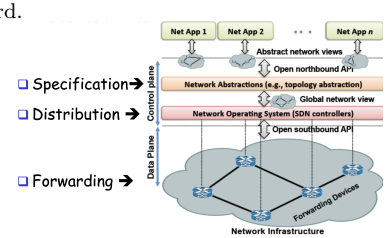


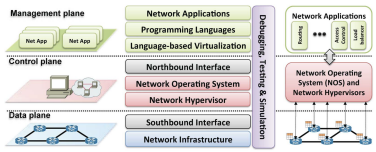
Separating the control and data planes makes network programming more robust and easier to use and diagnose. Firewall adds devices to enable additional function. Data plane is just a switch for SDN. Key is to find abstraction.
Specification. Allows control app to express desired network behavior or spec without implementing. Tells control plane what you want (ONOS).
Distribution. Shields SDN apps from distributed states, makes distributed control logically centralised.
Forwarding. Allows forwarding behavior desired by

apps while hiding details.
Network OS. Contol plane. Makes it easier to implement network control functionality. Achieve goal by writing apps rather than protocols based on certain forwarding plan.



SDN != OpenFlow. SDN is just an abstraction to make the entire data and control plane more robust. OpenFlow is the instruction for communication or a standard.

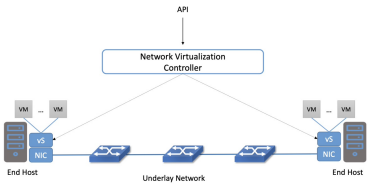




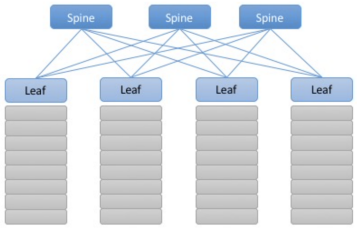
Cloud provider has 1 network. Virtualise physical network so user have impression they have their own. Done by isolating network namespace which is done by SDN in control plane.

SDN Examples

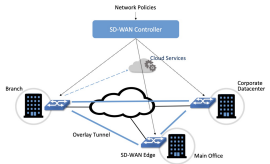
Network Virtualisation. Done by cloud provider, example is Docker. Resource orchestration framework would be k8s. VPN is a method to create virtual network by virtualising address. Attempts to ioslate the packets. Virtual switch on end hosts for virtual networks. Networks to be programmatically managed without manual configs. Disaggregation is a single API entry point to deal with virtual networks. Virtual networks have their own private address space.



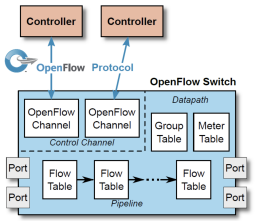
Switching Fabrics. Done by cloud provider. Quite similar to SDN. Leaf and spine graph are routers while the rack of servers are the switches. Important to manage cross machine traffic. Route and re-routing can achieve load balancing and latency. Rack-to-rack flows from server to leaf to server. Can use spine to help with the path.



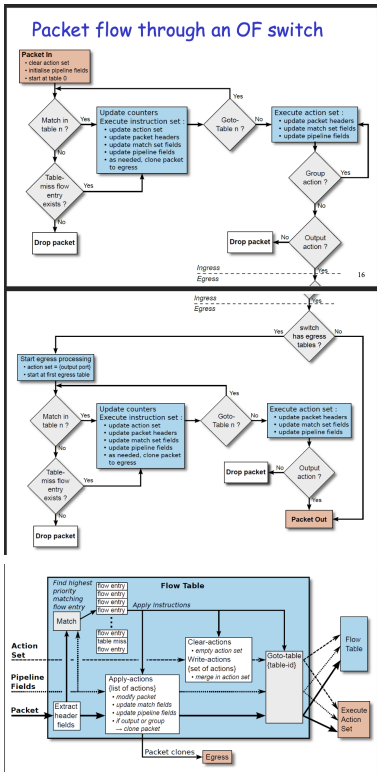
Traffic Engineering. Nodes are datacenters that can do things independently. If organised by SDN, can use their own SSSP. More optimal on global scale.
Software defined WANS. Enterprise solution. Doesn't rely on other AS and ISP. Owns the global backbone and uses SDN.



OpenFlow. Uses TCP. Single switch talks to multiple controllers. Virtualised to multiple switches.
OpenFlow Protocol. Open southbound API.
OpenFlow Switch. Forwarding abstraction that is northbound. Meter tables puts a meter on individual flows. Ingress and Egress to differentiate logic.



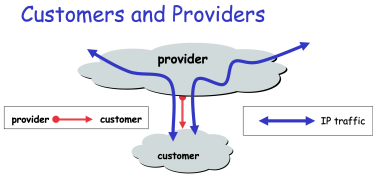
Flow Entry. Match fields are the fields that the packet is matched against. Contains header and pipeline fields. May be any (wildcarded) or subset of bits (bitmasked).
Pipeline Fields. Describes pipeline and propogate them. Records data as well.
Priority. Choose from multiple matches.
Instruction Set. Contains list of actions to apply immediately. Contains a set of actions to add to action set and to modify pipeline processing by going to another flow table. Default action if nothing matches.
Default. After table processing, go to enxt table. Cannot go back into pipeline. Possible to specify which table to go to next.
Timeout Behavior. Forward back to control plan since came from there anyway. Control plan try to write new things to switch. If no default entry then drop.



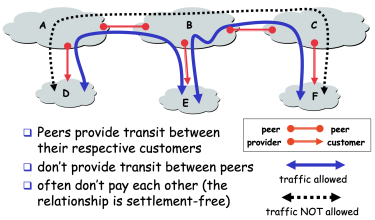
Extract header fields from the packet. To get a match, obtain the header and pipeline fields. Find the highest priority matching flow entry. Apply insturctions on the flow entry. Depending on the instruciton, we either apply, clear, or goto-table. The goto table tells us where the packet should go next for processing.
ONOS Model. Open northbound API. Specification abstraction.
ONOS System. Distribution abstraction.
Network OS. Any horizontally scalable cloud app. Consists of a set of loosely coupled subsystems, each handling an aspect and maintains its own service abstraction. Normally in a micro-service architecture, includes a scalable and highly available key-value store.
ONOS Architecture. Northbound interface is for apps to stay informed about network state. Use to control network data plane. Distributed core is responsible for managing network state and notifying apps about relevant changes in state. Internally a scalable key-value store. Southbound Interface is constructed from a set of plugins including shared protocol libs and device-specific drivers.
Abstractions. Intent is for network-wide topology-independent programming constructs. Flow objectives are finer-grained control, device centric programming constructs that are pipeline independent. Flow rules are use to control various pipelines, both fixed function and programmable.

Interconnection

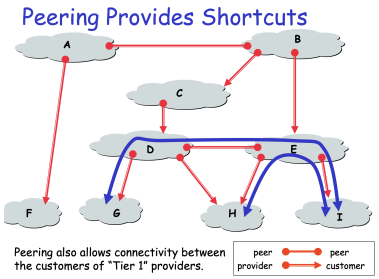
Autonomous System. Network of interconnected routers that are identified by unique AS number (ASN). Controlled by single admin domain. Company can have several ASN. Uses common routing protocol and policy.
Internet Structure. Small number of well-connected large networks. Tier-1 commerical ISPs provides national and international converage. Content providers like Google are private network that connects its data centers to Internet directly, bypassing T1 and regional ISP.
Internet Peering. Peering as a voluntary interconnection of adminstratively separated Internet networks to exchange traffice between users of each network. Bilateral agreements between neighbor AS - depends on biz relationships.



Customer pays provider to access internet and to reach everybody. A transit service is provided by provider.
Transit v NonTransit. Customer dont allow traffic to go through AS. If a network has multiple providers, then its called **multi-homing**. Traffic flow using a customer as a proxy for providers is not possible.
Selective Transit. AS can selectively provide transit service for its customers too.
Static Routing. Connect autonomous routing domain to Internet that is not through an AS.
P2P Relationship. Peer provide transit between **their respective customers** not between peers. Links are provided between customers.



B has no incentive to help since it doesnt gain anything from helping the customers of other peers. Its only being used by other peers to reach another peer. B will refuse helping since its not helping its own customers.



Peer when. Want to reduce upstream transit cost. Improve end-to-end performance. Only way to connect customer to internet.
Dont peer when. We rather have customers. Peers are competitors and negotiation is required.
Tier 1 AS. Peers with other T1s as 1 mesh. Doesnt pay for transit service since they are either

peers/customers. There will always be a link between 2 T1s.
Lower Layer. Provide transit to downstream customers but need at least 1 provider of their own.
Stub. Dont provide transit service. Connects to upstream providers.
Valley-free Property. Typically goes up to some AS, and the goes down to some customer. Single peak like a tree structure, where theres 1 root (go up and come down to another node). Single peer when theres only 1 other peer (go up, go to peer, go down). Any subpaths of the previous 2 are valid.
Validity check. Find an AS with an incentive not to help.
Internet Exchange Point. Bilateral peering which is a border router that connects to another domain. Neutral since it doesnt perform actions/prioritise particular ASes. Still need to pay for running infra. Only runs layer 2 and below (Data and Link). Can connect to other IXP switches. cheap peering by saving upstream costs. Keeps traffic local leading to better network performance, QoS and scalability.
Design. Each member brings a router and connects it to Ethernet switch or some other method as long as it connects to an Ethernet switch, which is the entry point for IXP.

BGP

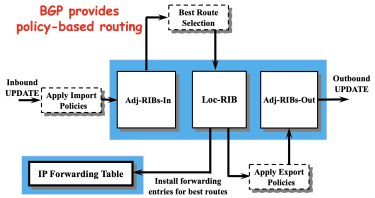
Challenges. Scale, privacy, policy (notion of cost which is not a universal definition). Traversing 1 link may be more ex than others. Depends on throughput and distance.
Link State Algorithm. All routers have complete topology. Each link has a cost. Every router has a global view. Use dijkstra. Open shortest path first.
Distance Vector Algorithm. Router communicates with neighbors. Gradually learn paths. Choose which path to use using an iterative process, ie Bellman Ford. Routing information protocol.
Limitation of link-state routing. Topology information is flooded which means high bandwidth and storage overhead. The nodes also divulge sensitive information leading to a security issue. Entire path computed locally per node which means high processing overhead in large network. Minimise some notion of total distance but works only if policy is shared and uniform.
Distance Vector Adv. Hides details of network topology. Only next hop is determined per node.
Disadvantage. Minimises some notion of total distance. Slow convergence.
Path-vector Routing. Extension of distance-vector routing with support for flexible routing policies. Advertise entire path. Since this is a path-vector, sends the entire path for each destination. Each node tells its neighbors how to reach d. So d : path(1) means that to get to d, we have to go to node 1. More nodes would be added to the left but read from left to right since that is the path from the current to the destination. Nodes can easily detect a loop by checking if itself is in the path. Simply discard paths with loops.
Border Gateway Protocol. Allow subnet to advertise its existence to reset of internet.
BGP Operations. Run by 2 routers from different As when they are directly connected. Establish TCP connection and exchange all active routes. As routes are being learnt, update neighbors as well.
eBGP. Exchanges infor from neighbor AS, implements routing policy. Should be directly connected, meaning no routers between them.
iBGP. propogate reachability info across backbone. Doesnt have to be directly connected. Assumption is that border router know where everyone is logically/physically. Must be (logically) fully meshed.
IGP. tells all border routers then propogates.
BGP Messages. OPEN - opens TCP connection to peer and auth sender. UPDATE - advertise changes in paths. KEEP ALIVE - keep connection alive in absence of UPDATE. NOTIFICATION - report error in previous message.
UPDATE msg format. Market is for encryption. Path attribute describe the path. Withdrawn routes are IP prefixes for the routes that were withdrawn.

Marker (16)	
Length (2)	Type (1)
Withdrawn Routes Length (2)	Withdrawn Routes (variable)
Path Attribute Length (2)	Path Attributes (variable)
Network Layer Reachability Information (variable)	

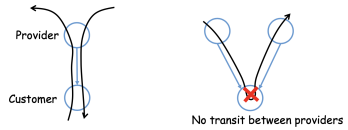
Withdrawn routes. RIP routes are given timers, if no update after time, assume no route. BGP rely on neighbors to tell them. All routers from a peer become invalid when peer goes down.
Path attributes. Well-known assumes every implementor recognises attribute. Transitive means forward to neighbor. Non-transitive is for AS to make their own rules.
Rules. Must recognise all well-known attribute. Mandatory attributes must be included in UPDATE messages. Once BGP peer updates well-known attributes, must pass them to peers.

Attribute Name	Category
ORIGIN	well-known mandatory
AS_PATH	well-known mandatory
NEXT_HOP	well-known mandatory
LOCAL_PREF	well-known discretionary
ATOMIC_AGGREGATE	well-known discretionary
AGGREGATOR	optional transitive
COMMUNITY	optional transitive
MULTI_EXIT_DISC (MED)	optional non-transitive

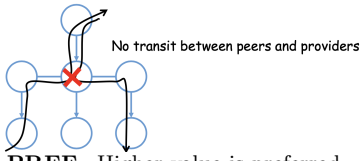
AS_PATH - sequence of number or set to known AS. NEXT_HOP where to go in neighboring AS. ORIGIN conveys origin of prefix learned from IGP (!, such as RIP/OSPF, interior gateway protocol) or EGP (e, exterior gateway protocol like BGP) or incomplete (?, unknown source).
Getting entries. Router becomes aware of IP prefix. Router determines output port for IP prefix. Router enters prefix-port pair in forwarding table. BGP message contains routes. A route is a prefix + attributes. Prefix is the destination ip.
AS_PATH. To reach destination, go through the sequence/set of AS from left to right.
NEXT_HOP. IP address of router interface that begins the AS_PATH.
Selecting best route. Router may receive multiple routes for same destination. Router needs to choose 1. Select based on shortest AS-PATH attribute. If there is a tie, find best intra-domain route.
Intra-domain route. Use selected route's NEXT_HOP attribute. Then use OSPF to find the shortest path from the router to the ip address at NEXT-HOP. Add prefix-port entry to its own forwarding table.
Hot Potato Routing. Deals with inter-domain routes. If there are multiple best ones, choose one with closest NEXT-HOP. Find the fastest way to get out of current AS.
Adding entries to forwarding table. Router become aware of prefix via BGP route advertisement from other routers. Determine router output port for prefix by choosing the best inter-AS route from BGP. Use OSPF to find best intra-AS route leading ot best inter-AS route. Identify port number for the best route. Enter prefix-port entry in forwarding table.
Routing Information Bases. A database of routes; all routes in a BGP speaker.
ADJ-RIBS-IN. Unprocessed routes from peers via inbound updates.
LOC-RIB. Local db and route that this AS use.
ADJ-RIBS-OUT. Selected for advertisement to peers.



Import policy. Filter unwanted routes from neighbor. Used to rank customer routes over peer routes. Manipulate attributes to influence path selection.
Export policy. Filter routes we dont want the neighbors to know, manipulate what they can see.
Customer Provider. NUS customer of Singtel. Provider advertise to everyone so traffic can reach NUS. Provider tell NUS how to access internet. NUS learn and propogate it to its customers/anything below NUS in network hierachy. Allow traffic to go in and out **from** its customers. Customer doesn't allow traffic between providers.



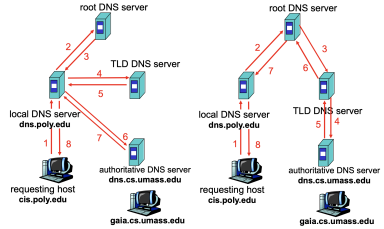
P2P. Singtel broadcase to peer, Comcast. Comcast forwards to its customers. Same rules apply for P2P relationships. Peers provide transit between respective customers. Dont provide transit between peers and (peers and provider).



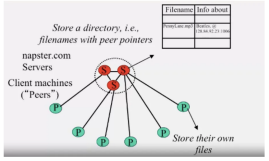
LOCAL_PREF. Higher value is preferred. Outgoing routing decision of internal AS. BGP speaker inform its other internal peers of its preference for a particular route includeded in UPDATE messages to internal, should not be shown to external.
MULTI_EXIT_DISC. NEXT-HOP is different. Preference affected externally. Choose smaller, logically is distance. Used by 1 AS to influence from other AS. Discriminate among multiple entry points to neighboring AS to control inbound traffic.
COMMUNITY. Associated with a set of routes. Any AS that see route and set community.
Best Route Selection. Calculate degree of preference. If learnt from internal peer, use LOCAL_PREF. Else, select highest degree of LOCAL_PREF, breaking ties by doing the following from left to right precedence: smallest number of AS numbers in AS_PATH, lowest origin number in ORIGIN, most preferred MED, router eBGP preferred over iBGP, lowest interior costs based on NEXT_HOP.
BGP Prefix Hijacking. Leads to blackhole (data traffic discarded), snooping(inspecting and redirecting), impersonation (send to bogus destination).
BGP subprefix hijacking. Originate a more specific prefix, traffic follows longest matching prefix.
Prevention. Each AS filter routes announced by customers.

P2P

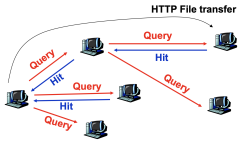
Model. Client-server is asymmetric. Delegation model adds new role of load balancing for server but client remains the same. Can be recursive or iterative.



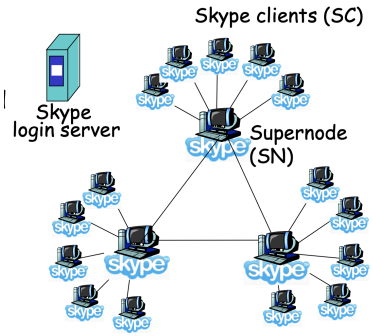
Properties. No always-on server or central entity. Arbitrary end systems directly communicate. No prior knowledge or structure. Flat architecture. Complete graph of hosts. Everyone is a client/peer, get info from others. Connected to other peers.
Problems. Peers are intermittenly connected and change IP addresses.
Napster. File sharing.
Method. Based on central index server. User register to become a peer. Send a list of index of files. Keep track of index of files that tells us which peer to download from. Server knows all the peers and files in network. Central index server is arranged in a distributed manner. Peers are just added to the edges.



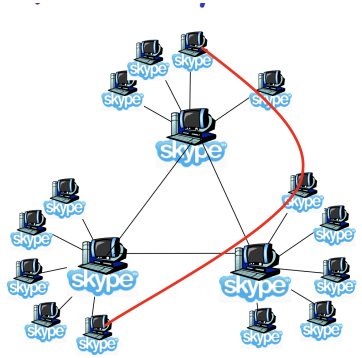
Strengths. Consistent view of network. Fast and efficient searching. Guarentees correctness of search as index of server can be trusted.
Weakness. Single point of failure of server. Large computations to handle queries. Downloading from 1 peer only. Unreliable content and vulnerable to attacks.
Gnutella. Pure P2P. Relies on a website where joining peers publish their identities. Flood network to find resource.



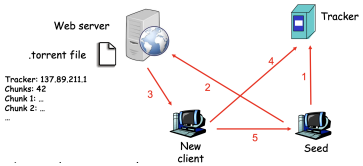
Strengths. Fully distributed. Open protocol; easy to write clients. Robust against node failures (only true for random failures). Less susceptible to DoS.
Weakness. Inefficient queries flooding; wastes alot of network and peer resources. Inefficient network management; constant probing.
KaZaA. Ordinary nodes for normal user peer. Supernodes for user peer with more resource/responsibilities. Top level only SN, low level only ON. ON belongs to 1 SN but can change from time to time. SN acts as a hub for its ON children. 1 SN can have many ON but 1 ON can only have 1 SN.
Supernode (SN). Exchange information between themselves. Dont form a complete mesh. Can change connections to other SN. Dont cache info from disconnected ON.
Ordinary Node (ON). Obtain address of SN, sends request and gives list of files to share. SN starts keeping track of ON. Not visible to other SN. Can be promoted if theres sufficient resource.
Skype. ALlows users to make calls to other computers on Internet. Similar architecture to KaZaA but skype is legal.



Pairs of users to communicate - inherently P2P. Proprietary, encrypted application layer protocol. Hierarchical overlay with SN. Index maps username to IP address that are distributed over SN.
Peers as relays. Problem when both Alice Bob, that are supernodes, are behind NATs, since NAT prevent outside peer from initiating a call to insider peer. Instead, a relay is chosen. Each peer initiates session with relay. Peers can then communicate through relay.



BitTorrent. Build a network for every file that is being distributed.
Strength. Send link to a friend. Link always refers to the same file. Not feasible on search-based as hard to identify particular files.
Weakness. No searching is possible. Websites exists but no name service.
Architecture. For each shared file, there is initially 1 server which hosts the original copy; the file is broken into chunks. Torrent file contains metadata about the content, hosted typically on a web server. Client downloads torrent file that indicates the sizes, checksums of chunks, and a tracker.



Seed starts the tracker. Seed creates file and host it somewhere. New client obtains torrent file. New client contacts tracker to obtain who has the file. New client download/exchange chunks with peers.
Torrent File. Table that tells us where tracker is and checksums of each chunk
Tracker. Tells us where peers are and the chunks checksums tells us whether we downloaded the same thing or not. Its a server that keeps track of which seeds and peers are in the swarm, doesnt participate in distribution.
Torrent. Group of peers exchanging chunks of file.
Swarm. Seeds + peers.
Details. Files are divided into 256kb chunks. Peer joining torrent has no chunks but will accumulate them over time. Register with tracker to get list of peers and connects to a subset of them. Peer uploads chunks to others so they can be downloaded. Peers can come and go but when they have entire file, they can choose to leave or stay.
Pulling Chunks. At any time, peers have different subset of file chunks. Peers periodically ask their neighbors for a list of chunks they have. Then they send a request for missing chunks from rarest to most common.
Pushing Chunks. Sends chunks to neighbors that are sending her at the highest rate; top 4 reevaluated every 10s. Every 30 seconds, randomly select another peer and send chunks to them. This attempts to find a higher upload rate to update its top4 providers. Newly chosen peer may join top4. When such a thing happen, the other peer can reciprocate.
Searching and Addressing. Finding objects depends on how network is formed, where objects are placed and how objects can be found efficiently. DNS is a mixture of local search and addressing with the root.
Addressing. Object location can be found efficiently. Each object is unique identifiable. Need to know unique name and maintain structure.
Searching. No need to know unique names; more user friendly. Hard to make efficient. Need to compare actual objects to know if they are the same. Just ask questions. Hidden costs. No definition of content, tell some kind of keyword to find. Problem arise if we dont know a suitable keyword.
Unstructured P2P. Searching Gnutella. Peers are free to join anywhere, choose neighbors freely and objects are stored anywhere.
Structured P2P. Allow for addressing, deterministic routing. Network structure determine where peers belong in the net and where objects are stored.
Key-Value Store. A hash table in the form of key-value pairs. Insertions, deletions and lookups are O(1). Hash function maps the keys to the buckets.
Distributed Hash table. Ech peer holds a portion of the table as the table is too big for 1 node. Each node is responsible for some buckets and as the nodes are being move about, the responsibilities changes. Communicate among themselves to find the responsible nodes. Supports all hash table operations.
Chord. Uses SHA-1 as hash function. Organised in a ring and the nodes keep track of predecessor and successor. To find successor go clockwise closest. To find predecessor go anticlockwise closest. Successors can be themselves.
Assigning Indices. Chord identifiers are represented using 160 bits. So identifier to each object is in the range 0 to $2^m - 1$, where m is the number of bits the identifier can be represented by. Assign indices to node that has the closest ID. Since convention is that the closest is the immediate successor.
Finding a node. Get successor of current node and count number of nodes to that successor.
Shortcuts. Reduce the number of steps to the destination, does not tell final destination. Table of shortcuts is at most size of namespace.Furthest shortcut from source cuts circle into half. Possible that there is no peer in the interval.
Node. Node is the first node that is greater than starting point of that entry. Denoted as first node $\geq n.finger[k].start$.
Interval. part of the namespace, start at a starting point and ends at the next starting point (exclusive) of the next shortcut. Denoted as $finger[k].start, finger[k+1]$
Starting point. starting point of shortcut. To calculate the starting point use $(n + 2^{k-1}) \bmod 2^m, 1 \leq k \leq m$, where n is the node number, 2^m is the number of nodes and k is the entry number starting from 1. Denoted $finger[k].start$
Node Join. When new nodes join the network, update that node in the shortcuts table of all the other nodes. Content mapped to other nodes moves there.
Routing. On query, hash the key and see which interval it is in. Then go to the node if the node is the successor we can stop, otherwise recursively use shortcuts until a nodde that is the successor is found. The number of steps for this is now log n.
Node leave. Require each node to keep track of its 2 successors instead. Periodically ping them to see if alive. Hash function makes things evenly distributed.
Performance. Finding an object takes log n steps. For N nodes and K objects, each node is responsible for $\frac{n}{k}$ objects. When a new node joins/leaves, responsibility of the last indice changes. Any node joining/leaving an N-node network uses $O(\log^2 n)$ messages to establish routing and finger tables and to initialise finger table and predecssors. Look up log$n$ tables for each log$n$ nodes.
CAN. Higher dimensional spaces in the shape of a 2D donut. Can take any higher dimensional space. Keep tracks of neighbors only. Dont need shortcuts. Number of neighbors in n-dimensions is $2n$.
Node join. Randomly generate coordinate. Route from an existing node to that coordinate to find which node owns that space. Split that space equally so that the new node can join.
Splitting/merging namespace. Split along x-axis first then y-axis and so on. Some neighbors merge back if possible. Some may need to handle other regions.
Routing. In 2D namespace, traversal search. Insertion requires a key-value pair. To get coordinates, use n number of hash functions.
Performance. Increasing number of dimensions increases size of routing table and number of hash functions but leads to a shorter path. State information is $O(d)$ since there are $O(2d)$ neighbors. Routing takes $dn^{\frac{1}{d}}$ with n nodes along each axis. Average path length is $\frac{d}{4}n^{\frac{1}{d}}$. In a circle, there are 2 halves and each half can go either clockwise or anticlockwise. $n^{\frac{1}{d}}$ is the number of nodes in each dimension.