Relógios

* nem todas as máquinas são acessíveis
* Computadores são baseados em quarts, ele vibra e conta as vibrações
* é impossível ter sincronização/precisão
* sincronização perfeita são os relógios lógicos ou analogicos(n entendi)
* slide dos dois relógios onde há dessincronização o problema é que quando fizer o próximo makefile o (.o) não será regenerado pq ele foi feito anteriormente e no registro conta a última alteração.
* um exemplo é o einstein com a teoria da relatividade, ele colocou que a gente pode ter formas diferentes de ver, percepções diferentes de ver, isso significa que nós em sistema modelamos a realidade e espelhamos a realidade, “causa e efeito”, nós não nascemos antes do pai.
* Tem que garantir algumas regras, sequências.
* Porque há mais de 1 processo? POrque é o conjunto de sistema s distribuídos
* Cada processo tem vários estados ao longo da vida. (Polini ta sonolento, se chamar ele vai para o estado zumbi)
* O que faz mudar o estado: Uma alteração no atributo (Local) ou recebeu um segmento de mensagem(Externo)/envio de mensagem.
* evento é a parte da alteração
* única forma de receber a mensagem é pela troca de mensagem Send/Receiver
* Um sistema centralizado não sabe quem enviou uma requisição primeiro mas ele resolve o problema de sequência, ele ameniza o problema.
* Skew: É a diferença de tempo livre entre dois relógios.
* Drift: Quanto meu relógio está longe a um erro do relógio papa.
* Tempo é uma abstração
* Sincronização externa: diferença entre meu relógio e o de um externo é menor que uma constante para todos os instantes
* Sincronização interna: diferença entre um instante é o instante seguinte é menor que uma constante para todos os instantes.
* Nunca há sincronização perfeita, porque o tempo que tu envia do tempo atual é o tempo + tempo de transferência.
* Tempo de transferência: é o RTT
* Problemas de RTT, posso ir por um caminho e voltar por outro
* Método de christian: Estima o erro de tempo
* Para utilizar o método, o sistema tem que aceitar a latência da rede
* Algoritmo de berkeley: Sincronização interna, tem que definir em conjunto o pc mestre, não considera cliente e servidor, teoricamente todos estarão sincronizados e se baseia ao método de christian
* Algoritmo de network time protocol: Mais famoso é uma aplicaçãozinha, baseado em UDP. Ele é uma árvore (Slide).

Replicação

* Replicar os recursos para atingir um objetivo
* Operação desconectada é quando tu ta no celular e ta sem rede mas consegue ver a notificação por estar previamente baixada
* Replicação é a chave por tolerar falhas e prover alta disponibilidade
* google decidiu replicar a tecnologia do que investir nela (Alta disponibilidade)
* disponibilidade é o tempo livre, sem operação desconectada
* melhoria de desempenho, um exemplo é proxie web
* maior disponibilidade: Como os dados estarão distribuidos, significa que se eu quero acessar um serviço e a maquina esta com problema, então poderá acessar outra maquina
* Melhoria de desempenho: DNS, servidores de cash, é na parte da paralelização de leitura, ela nos dá possibilidade de escalar ???
* tolerante a falhas: Se alguem falhar, outro sistema tem que estar funcionando
* Disponibilidade é o dropbox no pc ou modo offline do drive no celular por fazer replicação
* Dependabilidade: Ta so nos slides
* Replicação passiva é um backup, é o gerenciador primario, caso ele falhe os escravos serão promovidos (um deles)
* secundários são escravos

Coordenação e Acordos

* Possui 3 itens
* Exclusão Mútua: Não deve ser um problema, resolvido em sistemas operacionais.
* Problema é que em sop ocorre na mesma maquina não precisa se preocupar com memoria compartilhada, em sdi temos o problema de nao se baseara em variavel compartilhada
* protocolo para executar uma região crítica: enter, …(Slides)
* - algoritmo do servidor central: modo simples de cliente/servidor, servidor é autoridade maxima, ele verifica os recursos disponiveis. (Solicita, libera, concede)
* - Algoritmo de anel: 1→2→3...n→1→2, um anel, processo aponta para o seu vizinho. A operação funciona da seguinte maneira: vamos passar um bilhete para alguém , digamos que eu to com o cartão, talvez eu possa estar na usando a região crítica, se o processo estiver querendo entrar na região crítica, e eu receber o bilhete, entro na região crítica.
* - algoritmo usando multicast e relógio lógico:
* um processo dispara para todos se ele pode entrar na região critica, e talvez receba a resposta. Se eu receber a resposta ok de todos, eu entro, senão, significa que existe alguem querendo entrar e possui o menor relogico logico que o meu, mas só dos interessados (quero entrar, estou na regiao e recebi a mensagem).
* Algoritmo de votação: usa o mesmo princípio de MultiCast e relogio logico
* Problema do multicast é escalado, se estiver 1000 processos e 100 interessados, terá muitas mensagens.
* Nesse algoritmo, é por matematica, cada processo tem direito de um voto e quem estiver com maior voto, vai entrar na região crítica. Tem que ler nos slides.
* Eleição:
* Anel: Se dois processos chamarem eleição, então o cara manda o numero para seu vizinho (id) e vai passando, se algum quiser ser, verifica seus id, o maior será eleito, termina quando chegar nele mesmo.
* Algoritmo de eleição valentão: temos um conjunto de processos, p1, p2, p3, p4.
* P1 descobriu que p4 não estava mais coordenando.
* P1 pergunta para os pn com ids maiores do que ele para fazer eleição, caso p2 e p3 não aceitem, aí p2 envia eleição para p3 e p4. Por fim, p3 não obteve resposta do p4. Resumindo: Se todos recusarem, o que tiver o maior id ganha.
* Consenso: Problemas de acordo:
* A maioria tem que concordar com o resultado. Exemplo nos slides da transação de transferência.
* Quando não há consenso, é invalidado.
* Se houver o consenso, todos mudam. Exemplo de data de prova.
* Consenso em um sistema sincrono: Tem que ter x pessoas votando, para ocorrer o consenso
* Problemas dos generais bizantinos: Ninguem conhece ninguém, só há duas funções, ataca ou recua, eles só ganham se todos atacarem juntos. Traidores são os algoritmos que falham.

Web Service

* Seria a interface que faz comunicação entre cliente e servidor.
* Site de vendas fazem parte do web service.
* Você programa em algumas linguagens que tem suporte ao web service, mas não se programa em web service.
* korbann? Interoperabilidade: Para que tenhamos sistemas de sistemas, nao podemos fixar a forma de desenvolvimento. Nao pode obrigar a reprogramar tudo em uma nova linguagem que está na “moda”.
* o sucesso do web service está na visão de como se propagar a interoperabilidade.
* **ws nao pode ser utilizado por navegador.**
* Estudar o slide de infraestrutura com as camadas de aplicação

P2P

* Ganho de escala
* “Aplicativos exploram os recursos disponíveis nos limites da internet - armazenamento, ciclos de processamento, conteúdo presença humana”.
* Um indivíduo doando-se para o todo.
* Voce consegue colocar 32 nucleos no processador? Sim, mas estamos falando de alta escala, 1024 e aí não rola.
* Problema é organizar, alocar, pois, é um sistema descentralizado.
* Não se pode ter um disco com 100 terabytes e os outros tendo baixo, o desvio padrão deve ser pouco.
* Os nós não podem ficar ociosos
* Voce pode ter um sistema bancário P2P, só não pode ser somente P2P.
* Eles não são concorrentes, possuem nichos totalmente distintos.
* P2P só existiu porque na década de 90 pagava pela quantidade de uso. Assim, haviam pessoas que queriam fornecer dados, e assim surgiu P2P.
* Napster: Princípio de colaboração. Focado em mp3, musica
* Napster guardava o indice
* Napster era o arquivo inteiro, depois que tu tivesse baixado tudo, vc poderia compartilhar seus arquivos que acabou de baixar
* BitTorrent se tu baixou uma parte, tu já poderia ser compartilhado, e tu não baixa só de uma pessoa, são de várias.