# Zadání

* funkcionalita semestrální práce (chat, sdílená proměnná, …) = chat
* typ problému, který bude úloha řešit (volba vůdce, ukončení výpočtu, …) = volba vůdce
* implementovaný symetrický algoritmus (CHang-Roberts, …) = Chang-Roberts
* programovací jazyk a co se bude používat k transportu zpráv (Java RMI,
* C++ CORBA, Java JMS, Python sockety, …) = C#, .Net Core 3.0, gRPC

# Konfigurace

* 5 Virtuálních počítačů
* Privátní síť
* Vagrantfile – definice prostředí virtuální sítě a počítačů

# Topologie

* Každý uzel zná svoji adresu a své pořadí ve vektorových hodinách
* Každý uzel zná svého souseda
* Každý uzel zná souseda svého souseda

# Uspořádání událostí

* Každý uzel má vektorové hodiny
* Při přijetí zprávy aktualizuje hodiny
* Před odesláním zprávy zvýší čas na svých hodinách
* Každá zpráva obsahuje v hlavičce vektorové hodiny a unikátní identifikátor

# Přihlášení do systému (SignIn)

* Pokud se chce uzel přihlásit do systému, odešle libovolnému uzlu, který již v systému je svoji adresu a adresu svého následníka (typicky adresa uzlu, kterému zprávu posílá)
* Následně čeká až obdrží tuto zprávu, která obsahuje adresu jeho následníka a adresu následníka svého následníka
* Pokud se stane, že uzel se pokusí o chybné zapojení (náhodně vypadne a výpadek není detekován), uzly to detekuji a pošlou mu opravu zapojení

# Odhlášení ze systému (SIgnOut)

* Uzel pošle zprávu o odhlášení, která obsahuje adresu, následníka a následníka svého následníka
* Ostatní uzly na to reagují, pokud se odhlášený uzel týká jejich zapojení

# Výpadek ze systému (Dropped)

* Pokud uzel nečekaně přestane pracovat
* Uzly to zjistí při první posílané zprávě
* Pokud uzel detekuje výpadek svého následníka, informuje o tom následníka svého následníka
* Cluster zjistí kompletní informace o uzlu, jeho adresu, adresu jeho následníka a následníka svého následníka
* Jakmile jsou údaje shromážděny, reagují stejně jako v případě validního odpojení