
APRESENTAÇÃO

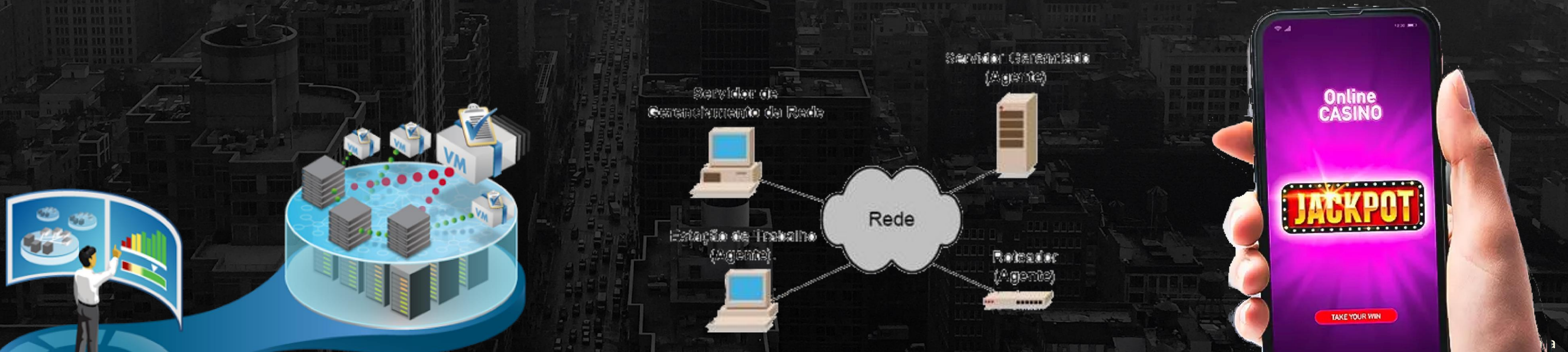
DETECTORES DE DEFEITOS MODELO HEARTBEAT

INTRODUÇÃO

- OBJETIVO: NESTA APRESENTAÇÃO, VAMOS EXPLORAR O CONCEITO DE HEARTBEAT, SUA IMPORTÂNCIA PARA A MANUTENÇÃO DE SISTEMAS DE ALTA DISPONIBILIDADE E COMO ELE É APLICADO EM DIVERSOS CONTEXTOS.
- IMPORTÂNCIA: EM SISTEMAS CRÍTICOS, A DETECÇÃO E A RESPOSTA RÁPIDA A FALHAS SÃO VITAIS PARA GARANTIR QUE OS SERVIÇOS PERMANEÇAM DISPONÍVEIS E CONFIÁVEIS.
- APLICAÇÕES: UTILIZADO EM CLUSTERS DE SERVIDORES, BALANCEADORES DE CARGA, REDES DE TELECOMUNICAÇÕES E OUTROS SISTEMAS QUE REQUEREM ALTA DISPONIBILIDADE.

O QUE É HEARTBEAT?

- HEARTBEAT É UM MECANISMO DE MONITORAMENTO CONTÍNUO QUE ENVIA SINAIS PERIÓDICOS (OU 'BATIDAS DO CORAÇÃO') ENTRE COMPONENTES DE UM SISTEMA PARA CONFIRMAR SUA OPERACIONALIDADE.
- SERVE PARA DETECTAR FALHAS RAPIDAMENTE E ACIONAR PROCEDIMENTOS DE RECUPERAÇÃO AUTOMÁTICA.



FUNCIONAMENTO DO HEARTBEAT

- COMPONENTES DO SISTEMA ENVIAM HEARTBEATS REGULARES PARA UM MONITOR OU ENTRE SI.
- SE UM HEARTBEAT ESPERADO NÃO É RECEBIDO DENTRO DE UM TEMPO PREDEFINIDO, É CONSIDERADA UMA FALHA E MEDIDAS CORRETIVAS SÃO ACIONADAS.

```
1 import threading
2 import time
3
4 # Intervalo de batimentos cardíacos (em segundos)
5 HEARTBEAT_INTERVAL = 2
6 # Limite de detecção de falhas (em segundos)
7 DETECTION_THRESHOLD = 5
8 # Número de heartbeats antes de forçar uma falha
9 FORCE_FAILURE_AFTER = 5
10
11 class HeartbeatDetector:
12     def __init__(self):
13         # Evento para indicar se um heartbeat foi recebido
14         self.heartbeat_event = threading.Event()
15         # Controle de execução dos loops
16         self.running = True
17         # Contador de heartbeats enviados
18         self.heartbeat_count = 0
19
20     def start(self):
21         # Inicia o envio de heartbeats
22         self.start_heartbeat()
23         # Inicia a detecção de falhas
24         self.start_detection()
25
26     def start_heartbeat(self):
27         def send_heartbeat():
28             # Continua enviando heartbeats enquanto o detector estiver em execução
29             while self.running:
30                 time.sleep(HEARTBEAT_INTERVAL)
31                 self.send_heartbeat()
32
33         # Cria e inicia uma thread para enviar heartbeats
34         heartbeat_thread = threading.Thread(target=send_heartbeat)
35         heartbeat_thread.daemon = True
36         heartbeat_thread.start()
37
38     def send_heartbeat(self):
39         if self.heartbeat_count < FORCE_FAILURE_AFTER:
40             # Simulação do envio de um heartbeat
41             print("Enviando heartbeat...")
42             # Define o evento indicando que um heartbeat foi enviado
43             self.heartbeat_event.set()
44             # Incrementa o contador de heartbeats enviados
45             self.heartbeat_count += 1
46         else:
47             # Simulação de falha: não envio mais heartbeats
```

```
Enviando heartbeat...
Enviando heartbeat...
Heartbeat recebido. Sistema está saudável.
Enviando heartbeat...
Enviando heartbeat...
Enviando heartbeat...
Heartbeat recebido. Sistema está saudável.
Enviando heartbeat...
Enviando heartbeat...
Enviando heartbeat...
Enviando heartbeat...
Heartbeat recebido. Sistema está saudável.
Enviando heartbeat...
Enviando heartbeat...
Enviando heartbeat...
Heartbeat recebido. Sistema está saudável.
Erro forçado: heartbeat não enviado.
Erro forçado: heartbeat não enviado.
Falha detectada! Heartbeat não recebido.
Executando procedimentos de recuperação...
```


APLICAÇÕES DO MODELO HEARTBEAT

MONITORAMENTO DE SERVIDORES EM DATA CENTERS

- EM DATA CENTERS, O MODELO HEARTBEAT É UTILIZADO PARA MONITORAR A INTEGRIDADE DOS SERVIDORES. CADA SERVIDOR ENVIA PERIODICAMENTE MENSAGENS HEARTBEAT PARA UM SISTEMA DE MONITORAMENTO CENTRAL. SE O SISTEMA NÃO RECEBE UMA MENSAGEM DE UM SERVIDOR DENTRO DO TEMPO ESPECIFICADO, ELE CONSIDERA QUE O SERVIDOR PODE ESTAR INATIVO OU TER FALHADO, PERMITINDO AÇÕES CORRETIVAS RÁPIDAS.

SISTEMAS DE GERENCIAMENTO DE REDES

- EM REDES DE COMPUTADORES, O MODELO HEARTBEAT É USADO PARA MONITORAR SWITCHES, ROTEADORES E OUTROS DISPOSITIVOS DE REDE. AS MENSAGENS HEARTBEAT AJUDAM A IDENTIFICAR RAPIDAMENTE FALHAS EM COMPONENTES DE REDE, GARANTINDO QUE A CONECTIVIDADE SEJA MANTIDA E MINIMIZANDO O TEMPO DE INATIVIDADE.

APLICAÇÕES DE ALTA DISPONIBILIDADE

EM SISTEMAS QUE REQUEREM ALTA DISPONIBILIDADE, COMO SISTEMAS FINANCEIROS E DE COMÉRCIO ELETRÔNICO, O MODELO HEARTBEAT É CRUCIAL. ELE MONITORIZA A DISPONIBILIDADE DOS SERVIDORES DE

APLICAÇÃO E BALANCEADORES DE CARGA, GARANTINDO QUE OS SERVIÇOS PERMANEÇAM OPERACIONAIS MESMO EM CASO DE FALHAS DE HARDWARE OU SOFTWARE.

APLICAÇÕES DO MODELO HEARTBEAT

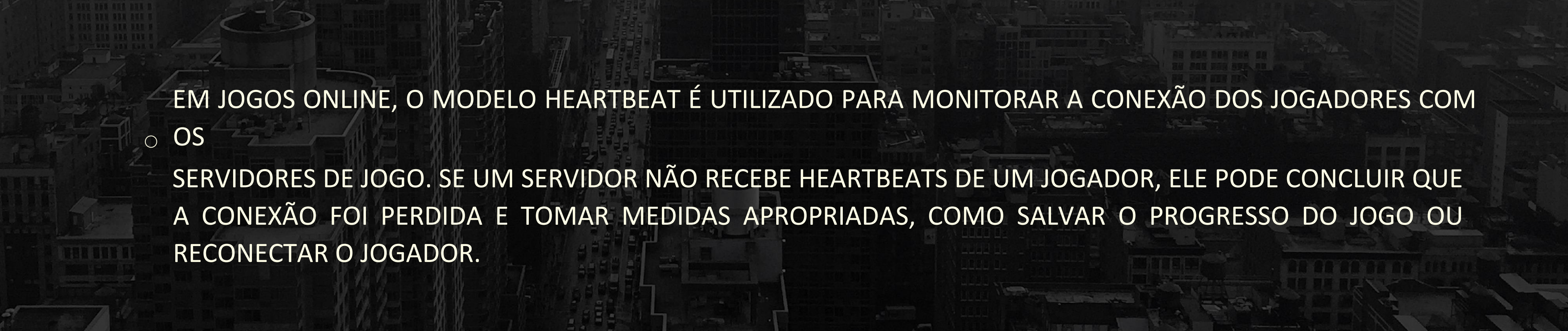
SISTEMAS DE CONTROLE INDUSTRIAL

EM AMBIENTES INDUSTRIAIS, O MODELO HEARTBEAT É APLICADO PARA MONITORAR EQUIPAMENTOS E SISTEMAS DE CONTROLE. SENSORES E CONTROLADORES ENVIAM HEARTBEATS PARA SISTEMAS DE MONITORAMENTO CENTRAL PARA GARANTIR QUE TODOS OS COMPONENTES ESTÃO FUNCIONANDO CORRETAMENTE. FALHAS DETECTADAS PODEM ACIONAR ALERTAS E PROCEDIMENTOS DE MANUTENÇÃO.

SISTEMAS DE INTERNET DAS COISAS (IOT)

NO CONTEXTO DA IOT, DISPOSITIVOS CONECTADOS ENVIAM MENSAGENS HEARTBEAT PARA INDICAR SUA OPERABILIDADE. ISSO É VITAL PARA A GESTÃO DE DISPOSITIVOS EM LARGA ESCALA, COMO EM CIDADES INTELIGENTES, ONDE SENSORES DE TRÁFEGO, ILUMINAÇÃO PÚBLICA E SISTEMAS DE VIGILÂNCIA PRECISAM SER CONSTANTEMENTE MONITORADOS.

APLICAÇÕES DE JOGOS ONLINE

- 
- EM JOGOS ONLINE, O MODELO HEARTBEAT É UTILIZADO PARA MONITORAR A CONEXÃO DOS JOGADORES COM OS SERVIDORES DE JOGO. SE UM SERVIDOR NÃO RECEBE HEARTBEATS DE UM JOGADOR, ELE PODE CONCLUIR QUE A CONEXÃO FOI PERDIDA E TOMAR MEDIDAS APROPRIADAS, COMO SALVAR O PROGRESSO DO JOGO OU RECONECTAR O JOGADOR.

EXEMPLO DE IMPLEMENTAÇÃO

- HAPROXY E KEEPALIVED:
- HAPROXY, UM BALANCEADOR DE CARGA, UTILIZA KEEPALIVED PARA GERENCIAR FAILOVERS.
- KEEPALIVED ENVIA HEARTBEATS PARA VERIFICAR A DISPONIBILIDADE DOS SERVIDORES PRINCIPAIS E DE BACKUP.

BENEFÍCIOS DO USO DE HEARTBEAT

- CONFIABILIDADE:
- AUMENTA A CONFIANÇA NA DISPONIBILIDADE DOS SISTEMAS AO GARANTIR MONITORAMENTO CONTÍNUO.
- RESPOSTA RÁPIDA A FALHAS:
- PERMITE A DETECÇÃO E A RESPOSTA RÁPIDA A FALHAS, MINIMIZANDO O TEMPO DE INATIVIDADE.
- MANUTENÇÃO PROATIVA:
- FACILITA A IDENTIFICAÇÃO PRECOCE DE PROBLEMAS ANTES QUE SE TORNEM CRÍTICOS, PERMITINDO MANUTENÇÃO PREVENTIVA.

DESAFIOS E CONSIDERAÇÕES

- LATÊNCIA E INTERVALO DE HEARTBEAT:
- O INTERVALO DE TEMPO PARA ENVIO DE HEARTBEATS DEVE SER CUIDADOSAMENTE AJUSTADO PARA EVITAR ALARMES FALSOS.
- CUSTO COMPUTACIONAL:
- A FREQUÊNCIA DE ENVIO DE HEARTBEATS PODE IMPACTAR O DESEMPENHO DO SISTEMA.
- CONFIGURAÇÃO E GERENCIAMENTO:
- REQUER CONFIGURAÇÃO PRECISA E MONITORAMENTO CONTÍNUO PARA SER EFICAZ.

OBRIGADO!!!

