Aula 4

IoT – Internet das Coisas

Prof. Gian Carlo Brustolin

1

Programação

2

Conversa Inicial

3

Programação para IoT

- Faremos agora uma abordagem um pouco mais aprofundada tecnicamente
- Não temos como objetivo mergulhar em códigos computacionais ou em programação de dispositivos

- Sistemas operacionais para IoT
- Computação orientada a serviços
- Arquitetura de micro-serviços reativos
- SDN e IoT

4

Introdução ao desenvolvimento para IoT

5

Introdução ao desenvolvimento para IoT

- Há "desenvolvimentos" em IoT
  - Complexidade crescente conforme simplificamos a camada sensorial
  - Desenho se altera conforme o tratamento parcial dos dados (*edge*, *fog*, *cloud*)

6

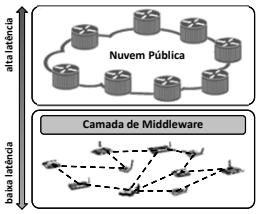
Introdução ao desenvolvimento: computação em nuvem e CoT

- A diversidade resultará na criação de vários ecossistemas sem integração
- As implementações de grande porte necessitam de funcionalidades para:
  - Gerenciar dados e dispositivos
  - Garantir entrega e o uso correto de dados
- Nuvem age como um transdutor e controlador

7

Introdução ao desenvolvimento: computação em nuvem e CoT

- CoT (Cloud of Things)
  - Conectividade com a nuvem
  - Os objetos se conectam diretamente
  - Demanda por largura de banda
  - Latência



Fonte: Santana *et al.*, 2019, p. 81.

8

Introdução ao desenvolvimento: computação de neblina e FoT

- Parte do processamento na rede local
- Aplicável, como arquitetura, a qualquer rede de computadores
- Evita levar todos os dados para a nuvem para selecionar os que devem voltar

9

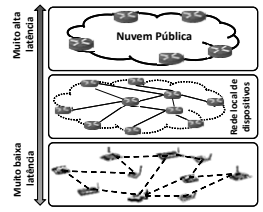
Introdução ao desenvolvimento: computação de neblina e FoT

- Inicialmente, em FoT (*Fog of Things*) o processamento é realizado por alguns dispositivos de maior capacidade pertencentes à rede de objetos
- Porém, o conceito de computação em neblina é distribuído, prevendo o uso da capacidade computacional onde ela for encontrada

10

Introdução ao desenvolvimento: computação de neblina e FoT

- FoT não elimina a necessidade da conexão à nuvem
- Presta os serviços locais possíveis
- Envia dados agendados e processados
- Baixa latência
  - Tempo real, mobilidade e escalabilidade



Fonte: Santana *et al.*, 2019, p. 82.

11

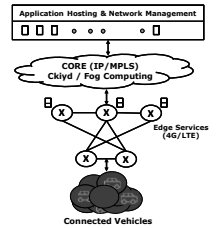
Introdução ao desenvolvimento: computação de borda

- *Edge computing* devolve a capacidade de processamento para os objetos
- Computação distribuída na borda
- Independência de conexão com a internet ou nuvem
- Baixa latência, mobilidade dos dispositivos e segurança de borda
  - VANET (*Vehicular Ad Hoc Networks*)

12

Introdução ao desenvolvimento: computação de borda

- ***Edge computing*** **permite:**
  - **Veículos inteligentes**
  - **Monitoramento de tráfego e semáforos inteligentes**
  - ***Smart grid***
  - **Monitoramento de tubulações de gás**



Topologia Edge para IoT (El-Sayed *et al.*, 2017, p.1709)

13

Introdução ao desenvolvimento: introdução à programação para IoT

- **O problema inicial são objetos de baixa interoperabilidade**
- **Ainda não existem padrões consolidados para as interfaces de *software***

14

Introdução ao desenvolvimento: introdução à programação para IoT

- **Devem ser mantidas em foco:**
  - **Gerenciamento de energia**
  - **Latência**
  - **Assincronicidade de TX**
  - **Metadados para depuração de dados na API**
  - **Ubiquidade**
- **Uso de microsserviços reativos**

15

Sistemas operacionais para IoT

16

Sistemas operacionais para IoT

- **Sistemas operacionais simples e leves e, em casos extremos, resumidos a temporizadores e transdutores**
- **Principais sistemas operacionais: Contiki e TinyOS, além do Android e algumas versões de Linux que podem ser orientadas à IoT**

17

Sistemas operacionais para IoT: Contiki OS e Contiki-NG

- **Opção a SOs rústicos (2006)**
- **Possui bibliotecas que permitem**
  - **Várias soluções de conectividade**
    - ✓ **Primeiro SO compatível IP V6 (µIPv6)**
  - **Alocação e controle de memória**
  - **Código extremamente enxuto**
    - ✓ **100 KB**
    - ✓ **10 KB de memória volátil**

18

Sistemas operacionais para IoT: Contiki OS e Contiki-NG

- Kernel em código aberto, utilizando a linguagem C
  - Orientado a eventos tornando
    - Eficiente em termos de energia e rápida em relação a eventos externos
  - Permite a conexão com APIs cooperativas e multitarefa
- Cooja (Contiki OS Java), que permite simular nós com memória e número de interfaces distintos

19

Sistemas operacionais para IoT: TinyOS

- Arquitetura em entidades computacionais independentes, ditas *componentes*, ligadas aos serviços oferecidos pelo objeto
  - Operam como classes de APIs

(...)

20

Sistemas operacionais para IoT: TinyOS

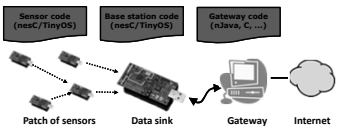
(...)

- Três tipos básicos:
  - Comandos (requisições para execução de serviço)
  - Tarefas (serviços internos do processador)
  - Eventos (que sinalizam o estado de um serviço)
- Existem bibliotecas de APIs que facultam a programação modular e reaproveitável

21

Sistemas operacionais para IoT: TinyOS

- Código aberto, baseado em nestC
- TOSSIM simulador, que permite exercitar o uso do TinyOS mesmo na ausência de HW
- Imaginado para IoT-A >3



IoT-A para sensores com TinyOS (Levis; Gay, 2009, p. 4)

22

Sistemas operacionais para IoT: TinyOS

- Foco em sistemas embarcados, LPWAN e redes de sensores
- Sensores extremamente simples conectados a um *gateway* com HW necessário aos sensores
- Consumo extra-baixo de energia na camada de percepção, adormecimento por até 99% do tempo

(...)

23

Sistemas operacionais para IoT: TinyOS

(...)

- Desenvolvimento feito em linguagens de mais alto nível e traduzidas para nestC – risco de baixa otimicidade
- Implantação em um nó, feita pelo *download*, com uso da interface USB

24

Sistemas operacionais para IoT: Android

- Não foi desenvolvido para objetos IoT
- Ambiente em torno do Android é estável e rico em opções
  - APIs, bibliotecas e mesmo *midlewares*
- Kernel Linux empilhado sob bibliotecas nativas do Android e uma máquina virtual (Android Runtime), que gerencia a execução das APIs
- Acima está o *framework* de serviços, que isola a base da pilha, fornecendo serviços de alto nível para os desenvolvedores

25

Sistemas operacionais para IoT: Linux

- Distribuições Linux adaptadas ou adaptáveis ao IoT:
  - Ubuntu Core – Baseada em *containers* com alta resiliência e boa segurança
  - RIOT – foco em IoT de baixo recurso
  - Raspian – Baseado no Debian: foco em uma placa IoT específica, a Raspberry

26

Computação orientada a serviços

27

Computação orientada a serviços

- Por que microsserviços reativos?
  - Introdução a SOC
  - Programação reativa
  - Microsserviços

28

Computação orientada a serviços: introdução a SOC

- *Service-oriented computing* – disponibilização, como serviços, de funcionalidades aplicacionais independentes
- Três camadas de funcionalidades:
  - Serviços propriamente ditos
  - Comunicação e agregação entre os serviços, sobresserviços: *web services*
  - Gerenciamento de serviços

29

Computação orientada a serviços: introdução a SOC

- Camada de comunicação e agregação permite a construção de macrosserviços compostos pelos serviços de base
  - *Web services* SOAP (*Simple Object Access Protocol*)
    - ✓ Interações por chamadas entre os serviços de base formatadas em XML

    (...)

30

Computação orientada a serviços: introdução a SOC

(...)

- *Web Services Representational State Transfer* (REST)
  - Métodos padronizados (PUT, POST, GET e DELETE) de troca de mensagens, entre serviços, utilizando HTTP
  - *Stateless* (não há troca ou armazenamento de contexto, que permanece no cliente)

31

Computação orientada a serviços: programação reativa

- *Functional Reactive Programming* foi criada para controle de robôs
- É uma estratégia de programação SOC que leva em conta os fluxos de dados durante a operação do código
  - Serviços eficientes, responsivos e elásticos, estáveis mesmo sob alta densidade de requisições

32

Computação orientada a serviços: programação reativa

- Execuções assíncronas de serviços motivadas pelos fluxos de dados
- Passagens assíncronas de dados entre serviços
- Permite a estabilidade de aplicação, mesmo em ambientes pouco previsíveis
- Necessário dissociar transmissor e receptor de uma mensagem, criando-se um espaço de armazenamento virtual das mensagens

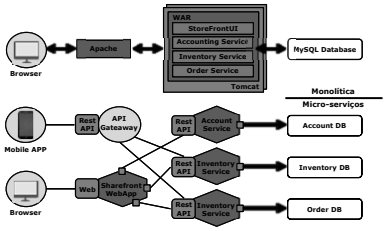
33

Computação orientada a serviços: microsserviços

- Microsserviço é o particionamento de uma API em uma coleção de pequenos segmentos de código independentes, que contenham um serviço ou parte segregável de um serviço
- Mecanismo leve de comunicação, ao estilo do HTTP; será implementado para permitir a conexão entre serviços, pelo armazenamento das mensagens necessárias a serem trocadas

34

Computação orientada a serviços: microsserviços



Arquitetura monolítica e microsserviços (Carrusca, 2018, p.10-11).

35

Computação orientada a serviços: microsserviços

- Microsserviço – responsividade e elasticidade maiores
- Desvantagem – performance (chamadas a serviços encadeados); latência alta
- Microsserviços reativos contornam esse problema pela atualização assíncrona dos dados, na camada de comunicação

36

Arquitetura de microsserviços reativos para IoT

37

Arquitetura de microsserviços reativos: comunicação

- A aplicação será composta por µS reativos, distribuídos em dispositivos e servidores localizados na borda da rede, na névoa ou na nuvem
- A comunicação poderá ser externa ou interna

38

Arquitetura de microsserviços reativos: comunicação

- Comunicação externa:
  - µS se comunica com um cliente da camada de agregação:
    - diretamente, serviço a serviço (*endpoint* público)
    - Por *gateway*, que fornecerá a comunicação também como um serviço
      - API para cada tipo de cliente, que agregará os dados necessários

39

Arquitetura de microsserviços reativos: comunicação

- Comunicação interna:
  - µS se comunica com outro µS:
    - Assíncrona, por *buffers*
      - Notificação (de um-para-um)
      - Publicação (um-para-muitos)
    - Síncrona – modelo pedido/resposta, mas essa estratégia é pouco prática para IoT

40

Arquitetura de microsserviços reativos: gestão de dados

- Uso de µS reativo pode ter como consequência queda na consistência dos dados
  - Um dado de uma amostragem anterior pode ainda estar presente quando se inicia uma nova coleta por uma API
- Solução: publicação reativa voltada a eventos
  - Um serviço somente publicará seus dados em caso de alteração em suas tabelas internas
  - Microsserviços, interessados nesses dados, vão subscreve-lo

41

Arquitetura de microsserviços reativos: descoberta de serviços

- *Service registry* – Componente de gerenciamento com a identificação e localização de cada µS
- Descoberta:
  - Contato direto com *service registry*
  - Uso de um *load balancer*

42

Arquitetura de microsserviços reativos: containers

- Boa estratégia para conter o uso de recursos computacionais de um microsserviço
- Concentração, em um único pacote, de todas as bibliotecas e definições para a execução de um serviço
- Especialmente útil em linguagens interpretadas
- Motor de container: Mesos Containerizer e Docker
- Orquestração: Kubernetes

43

SDN e IoT

44

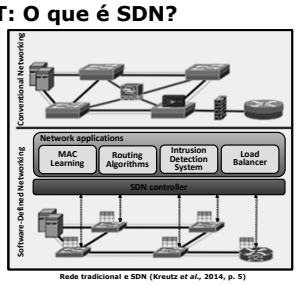
SDN e IoT

- A programação para dispositivos e redes IoT, elástica, responsiva e resiliente, pode ser bastante simplificada, se a rede puder sofrer adaptações dinâmicas

45

SDN e IOT: O que é SDN?

- ***Software defined networks*** **– O encaminhamento dos pacotes, ou plano de dados, é desassociado do controle da arquitetura da rede, ou do plano de controle**



Rede tradicional e SDN (Kreutz *et al.*, 2014, p. 5)

46

SDN e IoT: O que é SDN?

- A centralização do controle é apenas lógica
  - Redes complexas e distribuídas entre geografias
  - Tratam problemas de congestionamento e vulnerabilidades de segurança do TCP
  - Técnicas estatísticas, de ML, incorporam IA à rede, abstraindo completamente o plano de dados de seu gerenciamento humano
    - Predição de QoS e QoE (*quality of experience*), classificação de tráfego e gerenciamento dinâmico de segurança

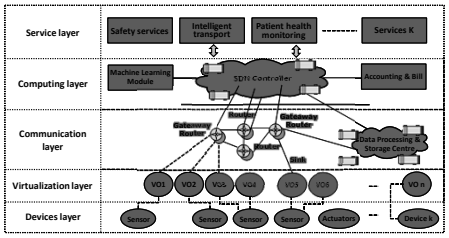
47

SDN e IoT

- ***Software-defined wireless sensor network*** **para IoT**
  - Contornar problemas de interoperabilidade, delegando o controle dos *gateways* a uma inteligência central
- Em cidades inteligentes, o trânsito de dados proveniente da borda em direção à nuvem é considerável e assimétrico

48

SDN e IoT



SDN em IoT (Ghaffar *et al.* 2021, p. 15)

49

Finalizando

- **Nesta aula, conhecemos os métodos recomendados de desenvolvimento de *software* para IoT, aventados para aplicações de larga escala**
- **Considerando a heterogeneidade das tecnologias em torno dos objetos IoT, as redes que os atendem precisam ser dotadas de boa dose de inteligência e flexibilidade**
- **Boa parte dessa heterogeneidade advém de protocolos particulares de conectividade**

50

51