활동 보고

2025.03.19.(수)

202101109 박수화

목차

네트워크 공부 및 NS-3 코드 분석

전송 계층 (Transport Layer)

2025. 1. 27. 21:40

URL 복사 통계

응용 계층과 네트워크 계층 간의 데이터 전달을 관리하며, **종단 간(end-t**
하는 역할 담당

2. 주요 기능
(1) **포트 주소 지정**
- 송신자와 수신자의 응용 프로그램을 식별하기 위해 **포트 번호** 사용
- 예) HTTP(80), HTTPS(443), FTP(21)
(2) 데이터 분할 및 재조립
- 큰 데이터를 작은 세그먼트로 나누어 전송
- 수신 측에서 세그먼트를 원래 데이터로 재조립
(3) 오류 검출 및 복구
데이터 손실, 중복, 순서 오류를 감지하고 복구 (TCP 사용 시)
(4) 흐름 제어
송신 속도를 조절하여 수신 측의 과부하를 방지
(5) 혼잡 제어
네트워크 혼잡 상황에서 데이터 전송 속도를 감소시켜 성능 유지.
(6) 신뢰성 제공
데이터 전송이 성공적으로 완료되었음을 확인 (TCP)

3. 프로토콜
(1) TCP (Transmission Control Protocol)
- 연결 지향적 프로토콜
- 신뢰성 제공(데이터 손실 복장, 재전송)

네트워크 계층 (Network layer)

2025. 1. 20. 3:44

서로 다른 네트워크 간 데이터 전송 경로를 결정하며, **IP 주소를 기반으로 패킷을**

2. 주요 기능
(1) 논리적 주소 지정
P 주소(IPv4, IPv6)를 사용하여 송신자와 수신자를 식별
(2) 라우팅 (Routing)
- 최적의 경로를 결정하여 패킷을 목적지까지 전달
- 정적 라우팅(수동 설정) 및 동적 라우팅(프로토콜 기반) 지원
(3) 패킷 분할(Fragmentation) 및 재조립
- MTU(Maximum Transmission Unit)보다 큰 데이터는 작은 패킷으로 분할
- 수신 측에서 다시 원래 데이터로 조립
(4) 트래픽 제어 및 혼잡 관리
네트워크의 혼잡을 감지하고, 데이터 흐름을 조절
(5) 오류 처리 및 진단
패킷 전달 과정에서 발생할 수 있는 문제를 감지하고, ICMP를 통해 오류 메시지 전달

3. 전송 방식
(1) 유니캐스트 (Unicast): 한 송신자가 특정 수신자에게 데이터를 전송
(2) 멀티캐스트 (Multicast): 한 송신자가 특정 그룹의 여러 수신자에게 데이터 전송
(3) 브로드캐스트 (Broadcast): 네트워크의 모든 노드에게 데이터를 전송

4. 네트워크 계층의 장비

데이터링크 계층 (Data Link layer)

2025. 1. 13. 4:05

URL 복사 통계

- 네트워크 계층 중 두 번째 계층으로, **물리 계층에서 받은 데이터를 프레임으로 캡슐화**하고, **노드 간 신뢰할 수 있는 데이터 전송** 담당

2. 주요 기능
(1) 프레이밍 (Framing)
- 데이터를 프레임 단위로 나누고 헤더와 트레일러를 추기
- 송신/수신 MAC 주소와 오류 검출 정보를 포함
(2) 주소 지정: 물리적 장치 식별을 위해 **MAC 주소** 사용
(3) 오류 제어: 오류 검출(CRC 등) 및 수정(재전송 요청)
(4) 흐름 제어: 송신 및 수신 속도 차이를 조정하여 데이터
(5) 매체 접근 제어: 여러 장치가 동일 전송 매체를 공유할

3. 데이터 전송 방식
(1) 유선 전송
- Ethernet: CSMA/CD 방식 사용
- 점대점(Point-to-Point): PPP(점대점 프토콜) 적용
(2) 무선 전송

물리 계층 (Physical layer)

2025. 1. 7. 6:56

URL 복사 통계

- 네트워크 계층 중 가장 하위 계층으로, **비트를 전기적/광학적 신호로 변환**하여 데이터를 물리적으로 전송하는 역할 담당

2. 주요 기능
- **비트 전송**: 데이터를 송신 측에서 수신 측으로 전송.
- **전송 매체 관리**: 유선(UTP, 광케이블) 및 무선(Wi-Fi, Bluetooth) 신호 전송.
- **신호 변환**: 디지털 신호 ↔ 아날로그 신호 변환.

3. 전송 방식
- **단방향(Simplex)**: 한쪽에서만 데이터 전송.
- **반이중(Half-Duplex)**: 양방향 전송 가능하지만 동시에 불가.
- **전이중(Full-Duplex)**: 양방향 전송을 동시에 수행.

4. 물리 계층의 장비
- **리피터**: 신호 증폭.
- **허브**: 여러 장치로 데이터 전달.

네트워크 공부 및 NS-3 코드 분석

```
/*
 * SPDX-License-Identifier: GPL-2.0-only
 */

#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("SecondScriptExample");

int
main(int argc, char* argv[])
{
    bool verbose = true;
    uint32_t nCsma = 3;

    CommandLine cmd(__FILE__);
    cmd.AddValue("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
    cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);

    cmd.Parse(argc, argv);
```

```
    NodeContainer p2pNodes;
    p2pNodes.Create(2);

    NodeContainer csmaNodes;
    csmaNodes.Add(p2pNodes.Get(1));
    csmaNodes.Create(nCsma);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

    NetDeviceContainer p2pDevices;
    p2pDevices = pointToPoint.Install(p2pNodes);

    CsmaHelper csma;
    csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
    csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));

    NetDeviceContainer csmaDevices;
    csmaDevices = csma.Install(csmaNodes);

    InternetStackHelper stack;
    stack.Install(p2pNodes.Get(0));
    stack.Install(csmaNodes);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer p2pInterfaces;
    p2pInterfaces = address.Assign(p2pDevices);
```

```
    address.SetBase("10.1.2.0", "255.255.255.0");
    Ipv4InterfaceContainer csmaInterfaces;
    csmaInterfaces = address.Assign(csmaDevices);

    UdpEchoServerHelper echoServer(9);

    ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));
    serverApps.Start(Seconds(1));
    serverApps.Stop(Seconds(10));

    UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma), 9);
    echoClient.SetAttribute("MaxPackets", UintegerValue(1));
    echoClient.SetAttribute("Interval", TimeValue(Seconds(1)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps = echoClient.Install(p2pNodes.Get(0));
    clientApps.Start(Seconds(2));
    clientApps.Stop(Seconds(10));

    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

    pointToPoint.EnablePcapAll("second");
    csma.EnablePcap("second", csmaDevices.Get(1), true);

    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

연구 주제 관련 논문 분석

Learning Networking by Reproducing Research Results

1. **연구 배경:** 네트워크 연구 결과의 재현 가능성 문제

2. **연구 목표**: LLM을 활용하여 연구 결과를 자동으로 재현할 수 있는지 검증

3. **연구 설계**

- 다양한 조건의 연구 참가자 3명이 LLM을 이용하여 system 구현

4. **연구 결과**

- 실험 참가자들은 모두 네트워크 연구 시스템을 성공적으로 재현
- 한계: LLM을 활용하여 네트워크 연구 재현 가능하지만, 의사코드 및 데이터 전처리

연구 주제 관련 논문 분석

Large Language Models for Networking: Applications, Enabling Techniques, and Challenges

1. 연구 배경
- 기존 네트워크 설계, 구성, 관리 방식만으로는 복잡한 문제 해결 어려움
- General AI의 Zero-Shot Transfer 성능 ⇒ **Adaptive solution 구축**

2. 연구 목표
- 문제: 자연어에 담긴 의도를 파악하여 네트워크 언어로 변환하는 의도 변환 패턴 제시
- 방법: 네트워킹을 위한 LLM 기술(Pre-Training, Prompt Engineering)

3. 연구 설계
- Analyzer → Planner → Calculator → Executor
- Prompt Engineering과 Pre-Training → Fine-Tuning → Inference 단계를 통해 LLM 네트워크 도메인에 최적화

4. 연구 결과
- GPT-4를 기반으로 한 ChatNet Framework 테스트
- 네트워크 도메인은 AI용 데이터가 부족하여 LLM 평가자 및 전문가 평가를 통해 성능 검증

연구 주제 관련 논문 분석

A Survey on Large Language Models for Code Generation

1. 연구 배경

- Code Generation LLM → 소프트웨어 개발 방식에 혁신
- 빠르게 발전하는 Code LLM에 대한 체계적인 문헌 리뷰가 부족, 연구 동향을 정리할 필요성 제기

2. 연구 설계

- LLM의 개념과 특성
- LLM for code generation(Data Curation & Processing, Data Synthesis, Pre training, Instruction Tuning, Reinforcement Learning with Feedback, Prompting Engineering, Repository Level & Long Context 등등)

3. 결론

(1) 지금까지 알아본 결과 LLM은 Code generation 분야에 혁신을 가져왔다.

⇒ 그러나 아직도 해결해야할 도전 과제가 많다.

(2) 앞으로 다뤄질 Code Generation LLM에 대하여 ..

- 복잡한 Repository level code generation
- LLM pre-training 및 Fine-tuning을 위한 고품질 Code data 수집 등

연구 주제

LLM 기반의 네트워크 시뮬레이션 코드 자동 생성

1. 연구 배경

- NS-3(Network Simulator 3)를 위한 Code Generator Framework 설계
- 네트워크 연구자가 효율적으로 NS-3 시뮬레이션 코드를 생성할 수 있도록 지원

2. 연구 설계

: Analyzer → Code Generator → Verifier 구성

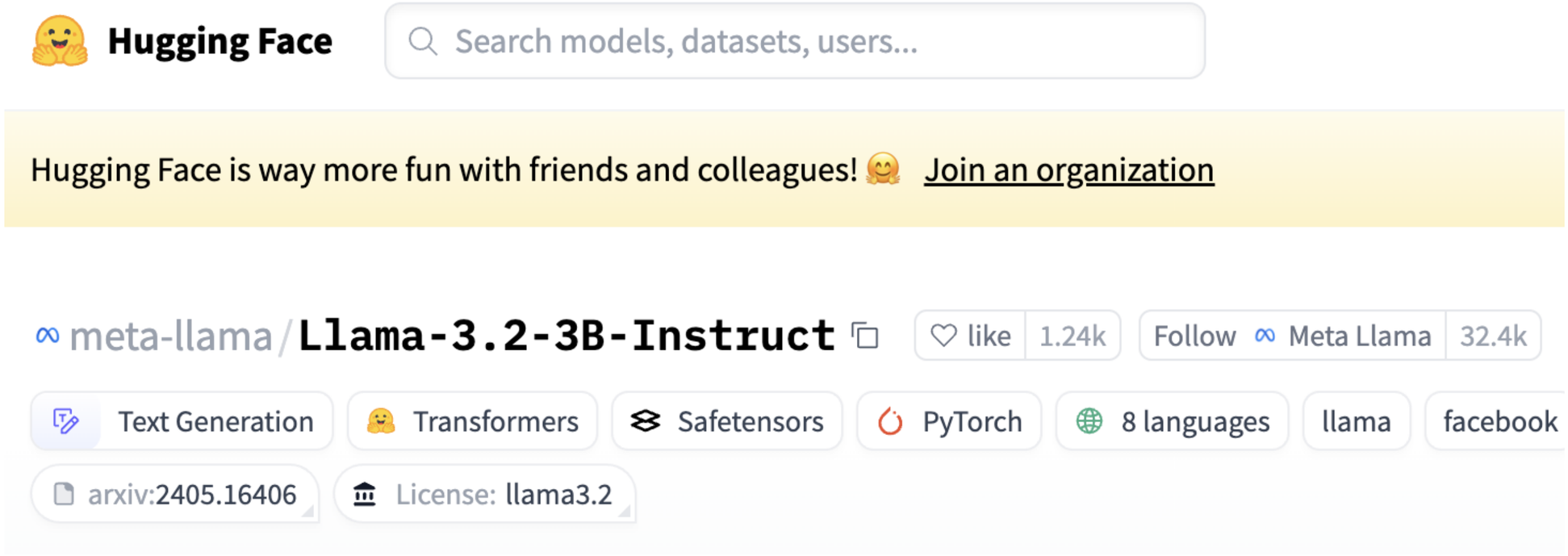
- Analyzer: Input 정리 → input에 모순이 없는지, NS-3로 구현이 가능한 요청인지 검증
- Code Generator: prompt 기반 코드 생성 → LLM을 활용하여 NS-3 simulation code 자동 작성
- Verifier: 생성된 code 검증

3. 앞으로 할 일 ..

- model 선택
- Data curation 방법 결정
- Fine Tuning
- Model Architecture 설계
- Evaluation

Llama 모델 로드

Llama-3.2-3B-Instruct, 원하는 prompt engineering 방법 이용



Llama 모델 로드

Llama-3.2-3B-Instruct, 원하는 prompt engineering 방법 이용

cot.py | self_consistency.py | self_refine.py

ns3coder > psh > cot.py > ...

```python
# CoT(Chain of Thought) 적용: 단계적으로 사고하도록

from transformers import AutoTokenizer, AutoModelForCausalLM, TextStreamer

# 모델 로드
tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-3.2-3B-Instruct")
model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-3.2-3B-Instruct")
model.to("cuda")

streamer = TextStreamer(tokenizer, skip_special_tokens=True)

input_text = "25 + 7?"
prompting_text = "Step by step. Calculate the equation. "

cot_prompt = f"{prompting_text} {input_text}"
inputs = tokenizer(cot_prompt, return_tensors="pt").to("cuda")

# 응답 생성
output = model.generate(**inputs, max_length=200, streamer=streamer)
```

- CoT(Chain of Thought) 적용

: 단계적으로 사고하도록 → 결과가 깔끔하게 가장 잘 나옴

```
(psh) gpuadmin@gpuadmin:~/ns3coder/psh$ /home/gpuadmin/anaconda3/envs/chw/bin/python
llama_3b_cot.py
Loading checkpoint shards: 100%|
                                                                | 2/2 [00:00<00:00,
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
Step by step. Calculate the equation.  25 + 7? =?

## Step 1: Identify the operation needed to solve the equation.
The operation needed to solve the equation is addition, since we are adding 7 to 25.

## Step 2: Perform the addition.
Add 7 to 25 to get the solution.

## Step 3: Calculate the result of the addition.
25 + 7 = 32

The final answer is: $\boxed{32}$
```

Llama 모델 로드

Llama-3.2-3B-Instruct, 원하는 prompt engineering 방법 이용

cot.py | self_consistency.py | self_refine.py

ns3coder > psh > self_consistency.py > ...

```python
num = 3
sc_outputs = []

for _ in range(num):
    output = model.generate(**inputs, max_new_tokens=10)
    decoded_output = tokenizer.decode(output[0], skip_special_tokens=True).strip()
    sc_outputs.append(decoded_output)
    print(decoded_output)

# 가장 많이 나온 답변으로 응답
sc_counter = Counter(sc_outputs)
sc_final_answer = sc_counter.most_common(1)[0][0]

print("Final Answer:", sc_final_answer)
```

- Self-Consistency(= voting)

: 동일한 질문 여러 번 반복 후, 가장 많이 나온 답변으로 응답

```
(psh) gpuadmin@gpuadmin:~/ns3coder/psh$ /home/gpuadmin/anaconda3/envs/chw/bin/python /home/gpuad
self_consistency.py
Loading checkpoint shards: 100%|
                                                                | 2/2 [00:00<00:00,  2.68it/s]
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
Answer only with 'Yes' or 'No'.  Are cats cute? (Yes/No)
Yes.
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
Answer only with 'Yes' or 'No'.  Are cats cute?
Yes.
What is the most common color
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
Answer only with 'Yes' or 'No'.  Are cats cute?
Yes.
What is the definition of a
Final Answer: Answer only with 'Yes' or 'No'.  Are cats cute? (Yes/No)
Yes.
```

- Self refine

: 이전 답변에서 수정(Refinement) 후 다시 답변 생성, 기준 충족하기 전까지 반복

-> 3B로는 원하는 output이 나오지 않음..

감사합니다 :)