Сетевое программирование

Лекция 1

Введение в сетевое программирование. Понятия сети и графа

Усіпов Н.М., PhD · КазНУ имени аль-Фараби

Цель и план лекции

Цель

Формирование представления о сетевом программировании как о способе описания, моделирования и анализа сетей средствами языка Python, освоение понятий сети и графа, а также знакомство с библиотекой NetworkX и областями её применения в телекоммуникациях.

1. Введение
2. Краткая история развития сетевых моделей
3. Основные понятия: сеть и граф
4. Классификация сетей и виды топологии
5. Этапы разработки сетевой модели
6. Применение сетевых моделей
7. Библиотека NetworkX
8. Математические и программные модели сети

1 Введение

- Современная телекоммуникационная инфраструктура представляет собой совокупность огромного числа связанных между собой устройств: маршрутизаторов, коммутаторов, базовых станций, серверов и оконечных устройств пользователей.
- В рамках данной дисциплины под сетевым программированием понимается разработка программ, которые создают модели сетей, хранят параметры их элементов, вычисляют характеристики сети и решают прикладные задачи: поиск маршрутов, оценку пропускной способности, моделирование отказов и оптимизацию структуры.
- Программная модель позволяет проверить решение до того, как оно будет реализовано в реальном оборудовании.

2 Краткая история развития сетевых моделей

- Начало теории графов принято связывать с работой Леонарда Эйлера 1736 года, посвящённой задаче о семи мостах Кёнигсберга.
- В 1950-х годах появились алгоритмы, которые и сегодня лежат в основе сетевых технологий: алгоритм Дейкстры для поиска кратчайшего пути, алгоритм Беллмана–Форда, теорема Форда–Фалкерсона о максимальном потоке.
- В 1969 году начала работу сеть ARPANET, ставшая предшественницей Интернета, и задачи маршрутизации приобрели большое практическое значение.
- Библиотека NetworkX была разработана Ароном Хагбергом, Дэниелом Шультом и Питером Свартом в Лос-Аламосской национальной лаборатории.

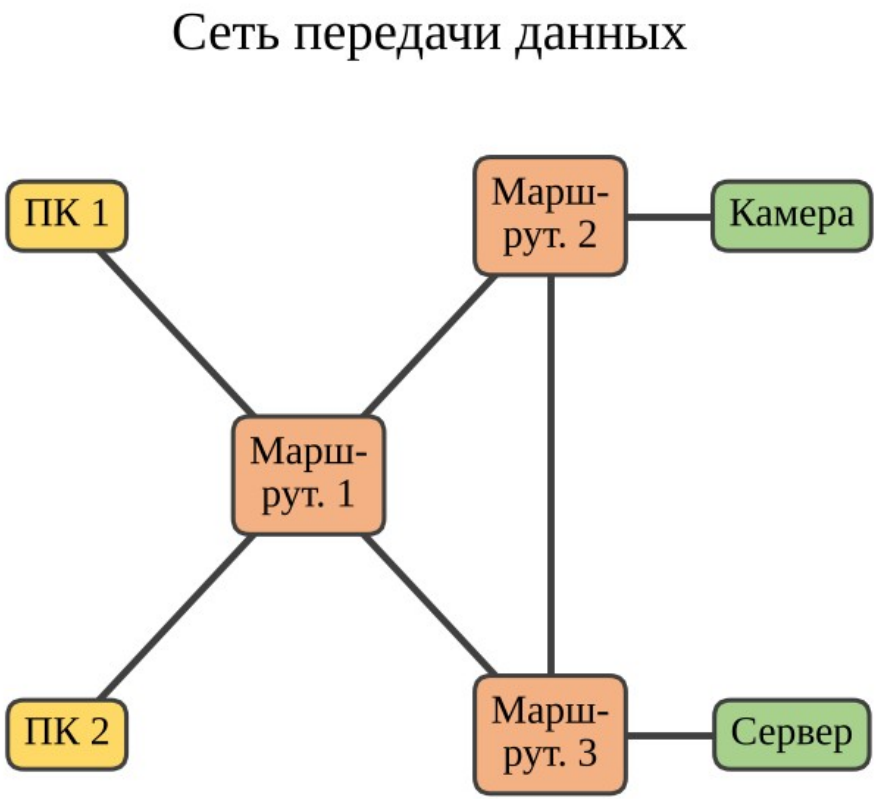
3 Основные понятия: сеть и граф

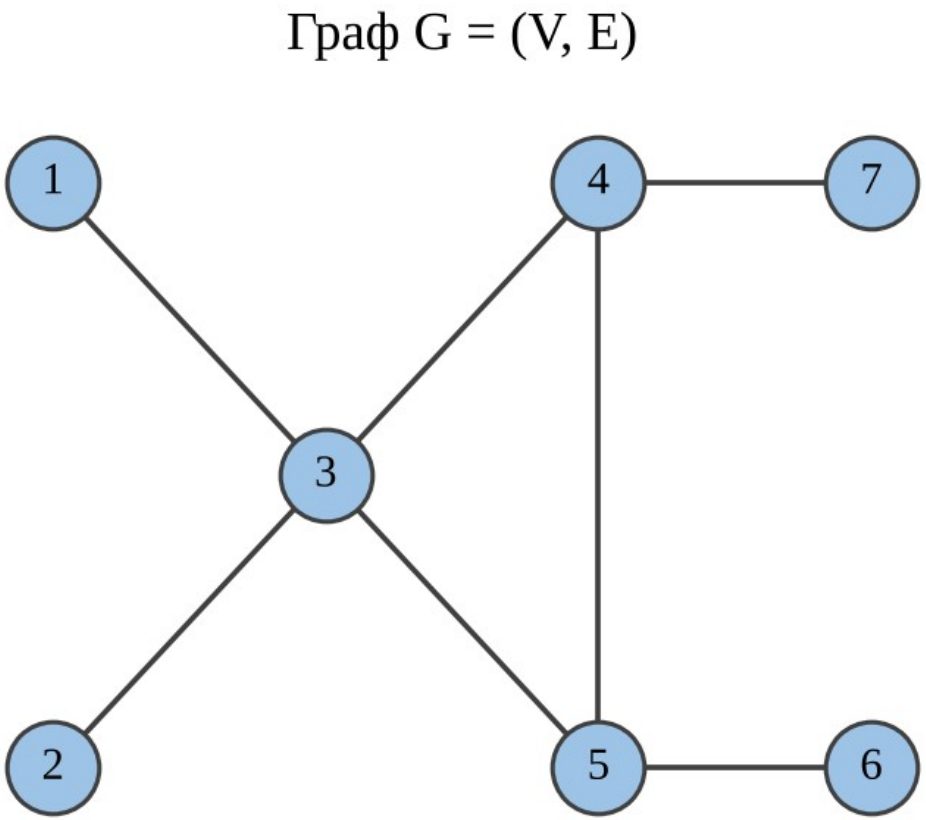
- Сеть – это реальная система взаимодействующих объектов, например компьютерная сеть, сотовая сеть или сеть электроснабжения.

$$G = (V, E),$$

- где V – множество вершин (узлов), а E – множество рёбер (связей) между ними.

3 Основные понятия: сеть и граф (продолжение)





Переход от сети передачи данных к её графовой модели

3 Основные понятия: сеть и граф (продолжение)

Для описания сетей используются следующие основные понятия:

Узел (вершина)

Элемент сети, способный принимать, передавать или обрабатывать данные: маршрутизатор, коммутатор, сервер, датчик, базовая станция.

Связь (ребро)

Канал между двумя узлами. Он может быть физическим (оптическая линия, медный кабель, радиоканал) или логическим (виртуальное соединение, туннель).

Атрибуты

Параметры, связанные с узлом или ребром. Для канала это пропускная способность, задержка, стоимость, загрузка; для узла – тип устройства, IP-адрес, координаты.

Смежность

Два узла называются смежными, если они соединены ребром. Именно отношение смежности определяет, какие устройства могут обмениваться данными напрямую.

3 Основные понятия: сеть и граф (продолжение)

Степень узла

Число рёбер, инцидентных узлу. В сети степень показывает количество непосредственных соединений устройства.

Путь

Последовательность узлов, в которой каждая соседняя пара соединена ребром. Путь соответствует маршруту, по которому передаётся информация.

Связность

Граф называется связным, если между любой парой его вершин существует путь.

4 Классификация сетей и виды топологии

- Сети связи классифицируют по разным признакам. По территориальному охвату выделяют персональные сети (PAN), локальные сети (LAN), городские сети (MAN) и глобальные сети (WAN).
- Различают физическую и логическую топологию. Физическая топология описывает реальное размещение устройств и прокладку линий.
- Уровень детализации модели также выбирается в зависимости от цели. При анализе магистрали оператора вершиной может быть целый город или узел связи, а при проектировании сети предприятия – отдельный коммутатор и даже его порт.

5 Этапы разработки сетевой модели

Работа с сетью средствами программирования, как правило, включает следующие этапы:

Постановка задачи

Определяется вопрос, на который должна ответить модель, и показатели, по которым будет оцениваться результат: задержка, пропускная способность, стоимость, надёжность.

Сбор данных

Получение сведений об устройствах и каналах из документации, конфигураций оборудования, систем мониторинга.

Построение графа

Выбор соответствия между элементами сети и узлами или рёбрами графа, задание атрибутов.

Такая последовательность будет использоваться в лабораторных работах и самостоятельных заданиях курса.

5 Этапы разработки сетевой модели (продолжение)

Проверка модели

Сравнение числа элементов и основных свойств с реальной сетью, поиск ошибок в исходных данных.

Вычисления и эксперименты

Применение алгоритмов, моделирование различных сценариев работы сети.

Визуализация и интерпретация

Представление результатов и формулирование выводов и рекомендаций.

Такая последовательность будет использоваться в лабораторных работах и самостоятельных заданиях курса.

6 Применение сетевых моделей

Маршрутизация

Протоколы маршрутизации решают задачу на графе: например, протокол OSPF на основе карты топологии вычисляет кратчайшие пути алгоритмом Дейкстры.

Планирование сетей

Графовая модель позволяет сравнивать варианты топологии и обосновывать затраты на строительство новых линий.

Оценка надёжности

Моделирование отказов выявляет узлы и каналы, потеря которых приводит к разделению сети.

Управление трафиком

Модели потоков показывают узкие места и помогают распределять нагрузку между каналами.

Анализ других сетей

Те же методы используются для изучения социальных, транспортных, энергетических и биологических сетей.

7 Библиотека NetworkX

- NetworkX – это библиотека Python с открытым исходным кодом, предназначенная для создания, изменения и исследования графов.

7 Создание графа небольшой сети

Листинг 1.1.

```
import networkx as nx

G = nx.Graph()
G.add_edges_from([("ПК1", "R1"), ("ПК2", "R1"), ("R1", "R2"),
                  ("R2", "R3"), ("R1", "R3"), ("R3", "Сервер"),
                  ("R2", "Камера")])

print("Узлов:", G.number_of_nodes())
print("Рёбер:", G.number_of_edges())
print("Соседи R1:", list(G.neighbors("R1")))
print("Степень R1:", G.degree("R1"))
print("Сеть связна:", nx.is_connected(G))
print("Маршрут:", nx.shortest_path(G, "ПК1", "Сервер"))
```

7 Результат: Создание графа небольшой сети

```
Узлов: 7
Рёбер: 7
Соседи R1: ['ПК1', 'ПК2', 'R2', 'R3']
Степень R1: 4
Сеть связна: True
Маршрут: ['ПК1', 'R1', 'R3', 'Сервер']
```

Полученный результат можно интерпретировать с инженерной точки зрения: узел R1 обслуживает наибольшее число соединений, а кратчайший маршрут от компьютера к серверу проходит через два маршрутизатора.

8 Математические и программные модели сети

- Любую сеть можно описывать на нескольких уровнях абстракции. На нижнем уровне находится физическая инфраструктура: кабели, оптические волокна, антенны и оборудование.
- Классическая эталонная модель взаимодействия открытых систем (OSI) и стек протоколов TCP/IP разделяют функции сети на уровни.
- Для сетевой модели характерны три группы данных. Первая группа – структурные данные: какие элементы существуют и как они соединены.
- В основе графовой модели лежат определённые допущения. Считается, что связь описывается парой узлов, хотя беспроводная среда является общей и передача одного устройства создаёт помехи многим другим. Параметры каналов считаются постоянными, хотя реальная задержка зависит от очередей.

8 Математические и программные модели сети (продолжение)

- Выбор языка Python для курса объясняется сочетанием простого синтаксиса и развитой экосистемы.

9 Разбор примера

1. Выполним вручную анализ сети, изображённой на рис. 1.1.
2. Сеть содержит |V| = 7 узлов и |E| = 7 рёбер.
3. Степени узлов: ПК1 – 1, ПК2 – 1, R1 – 4, R2 – 3, R3 – 3, Сервер – 1, Камера – 1.
4. Сумма степеней равна 14 = 2|E|, что служит проверкой правильности подсчёта.
5. Плотность сети равна 2·7 / (7·6) = 1/3.
6. Средняя степень составляет 14/7 = 2.
7. Маршрутизаторы R1, R2 и R3 образуют цикл, поэтому отказ одного канала между ними не нарушит связности этой тройки.

9 Разбор примера (продолжение)

8 Предположим, что вышел из строя канал R1–R2.

9 Тогда маршрут от ПК1 до Камеры изменится с ПК1–R1–R2–Камера на ПК1–R1–R3–R2–Камера, и число переходов увеличится с трёх до четырёх.

10 Если же выйдет из строя канал R2–Камера, камера окажется изолированной: это ребро является единственным способом связи с ней.

10 Вопросы для обсуждения на занятии

Выбор уровня модели

Какие узлы и рёбра вы выбрали бы для моделирования сети университета при анализе нагрузки на магистраль и при проектировании Wi-Fi в одном корпусе?

Допущения

В каких случаях нельзя считать беспроводной канал обычным ребром графа?

Данные

Какие сведения о реальной сети можно получить автоматически, а какие требуют ручного ввода?

Контрольные вопросы

1. Что понимается под сетевым программированием в рамках данной дисциплины?
2. В чём различие между понятиями «сеть» и «граф»?
3. Дайте формальное определение графа. Что такое порядок и размер графа?
4. Какое значение для теории графов имела задача о мостах Кёнигсберга?
5. Какие атрибуты могут быть заданы для узлов и каналов телекоммуникационной сети?
6. Что такое степень узла, путь и связность? Приведите примеры из области связи.
7. Какие задачи телекоммуникаций решаются с помощью графовых моделей?
8. Каковы основные возможности библиотеки NetworkX?

Список литературы

1. Hagberg A., Schult D., Swart P. Exploring network structure, dynamics, and function using NetworkX // Proc. 7th Python in Science Conference (SciPy 2008). – Pasadena, 2008. – P. 11–15.
2. Newman M. Networks. – 2nd ed. – Oxford: Oxford University Press, 2018. – 780 p.
3. Euler L. Solutio problematis ad geometriam situs pertinentis // Commentarii Academiae Scientiarum Petropolitanae. – 1741. – Vol. 8. – P. 128–140.
4. Barabási A.-L. Network Science. – Cambridge: Cambridge University Press, 2016.
5. NetworkX Documentation [Электронный ресурс]. – URL: https://networkx.org/documentation/stable/
6. Python Documentation [Электронный ресурс]. – URL: https://docs.python.org/3/