Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Mobile Computing Lab

Assignment 4

Ad-hoc Communication

Frank Dürr, Saravana Murthy, Zohaib Riaz, Ahmad Slo

Overview

- Motivation: Wireless Ad-hoc Networks require dedicated **ad-hoc routing protocols** to cope with dynamic network topology
  - In this assignment, we consider mesh networks
    - We will use a **real mesh network** deployed at IPVS
  - Considered routing protocols:
    - Flooding-based routing
    - Dynamic source routing

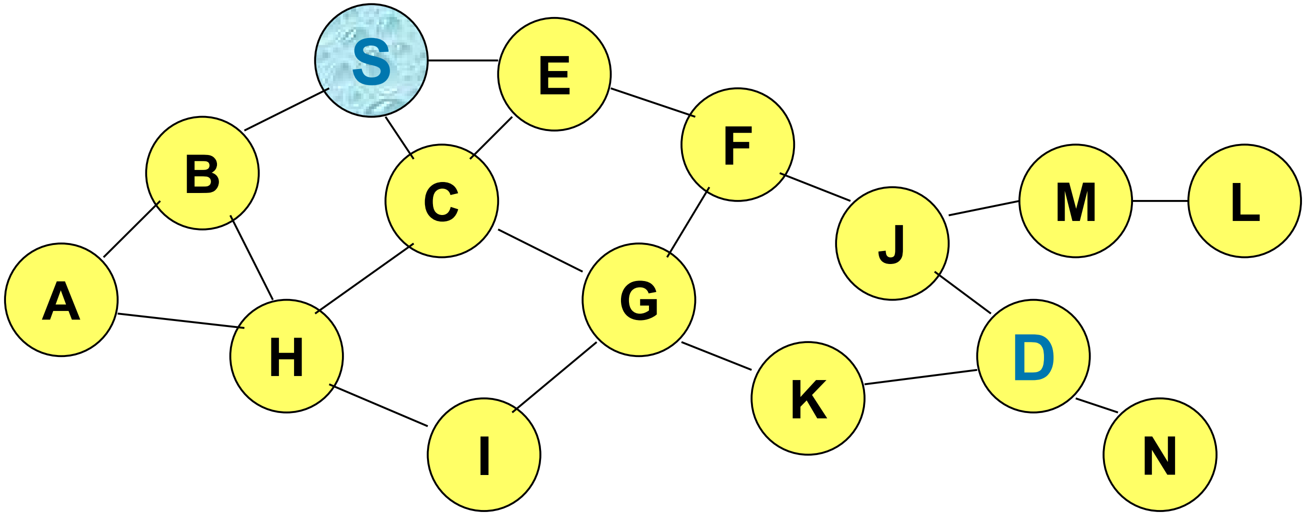
- Tasks:
  1. Implementation of Flooding
  2. Implementation of Dynamic Source Routing (DSR)

Task 1: Flooding

- Implementation of **Flooding-based routing protocol**
  - On application layer in Java
  - Communication via **UDP broadcast messages**

- Basic characteristics of Flooding
  - Simple routing protocol
  - Messages are forwarded to all neighbors
  - No additional control packets required
  - High network overhead (congestion, packet loss), but also high robustness

Flooding-based Routing: Example

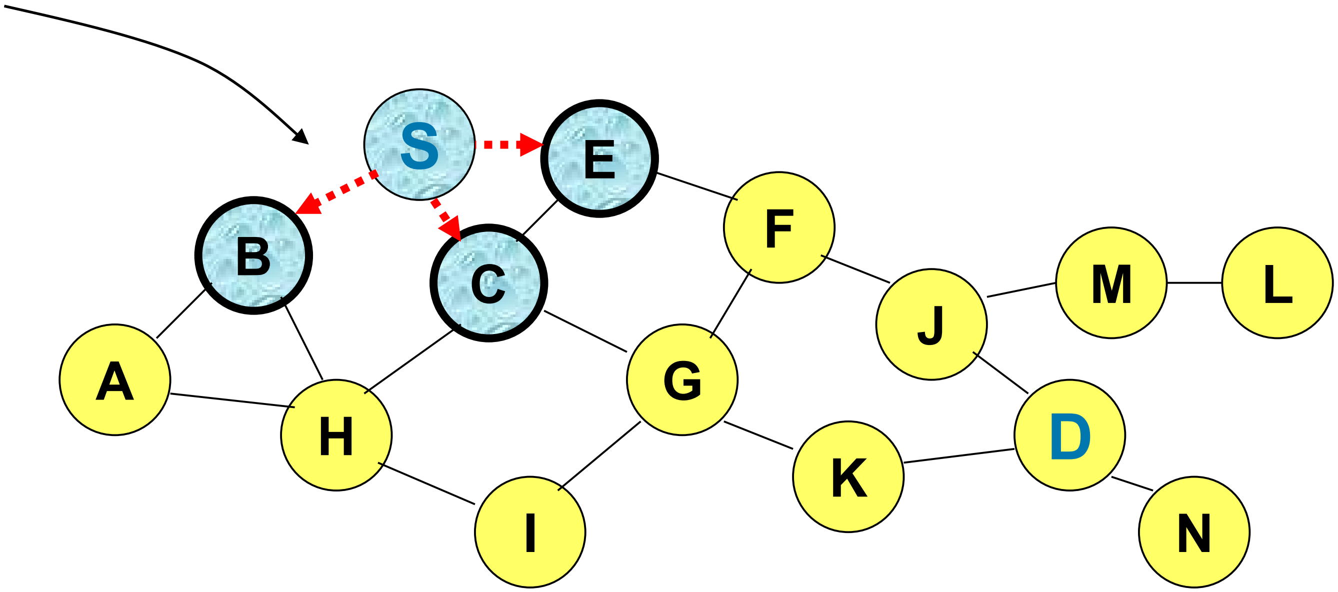


● Represents a node that has received packet P

— Represents that connected nodes are within each other's transmission range

Flooding-based Routing: Example

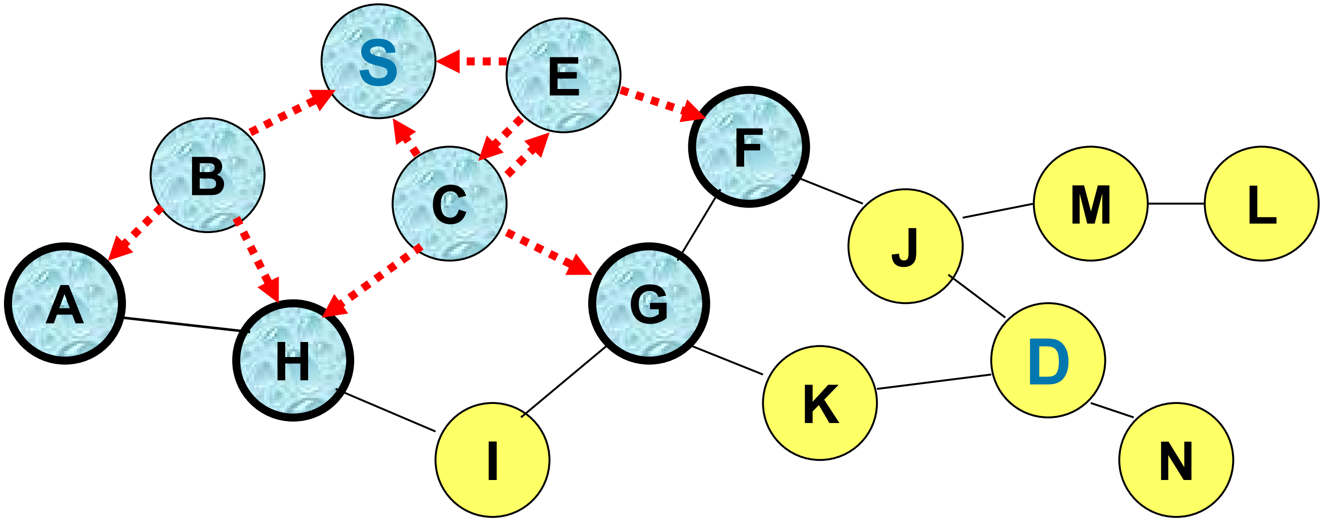
Broadcast transmission



◉ Represents a node that receives packet P for the first time

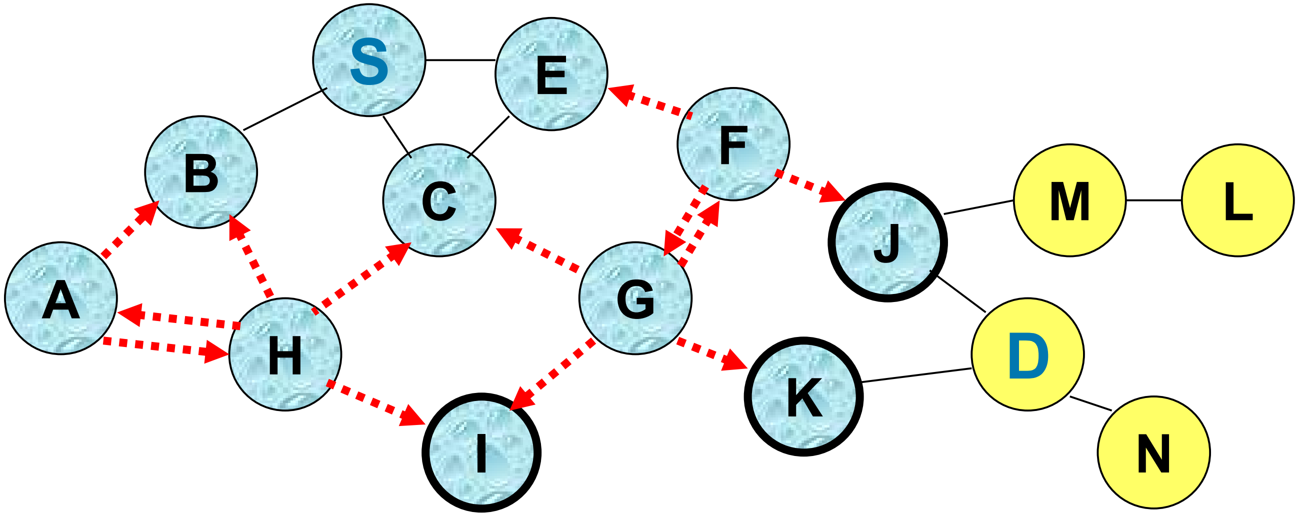
⇢ Represents transmission of packet P

Flooding-based Routing: Example



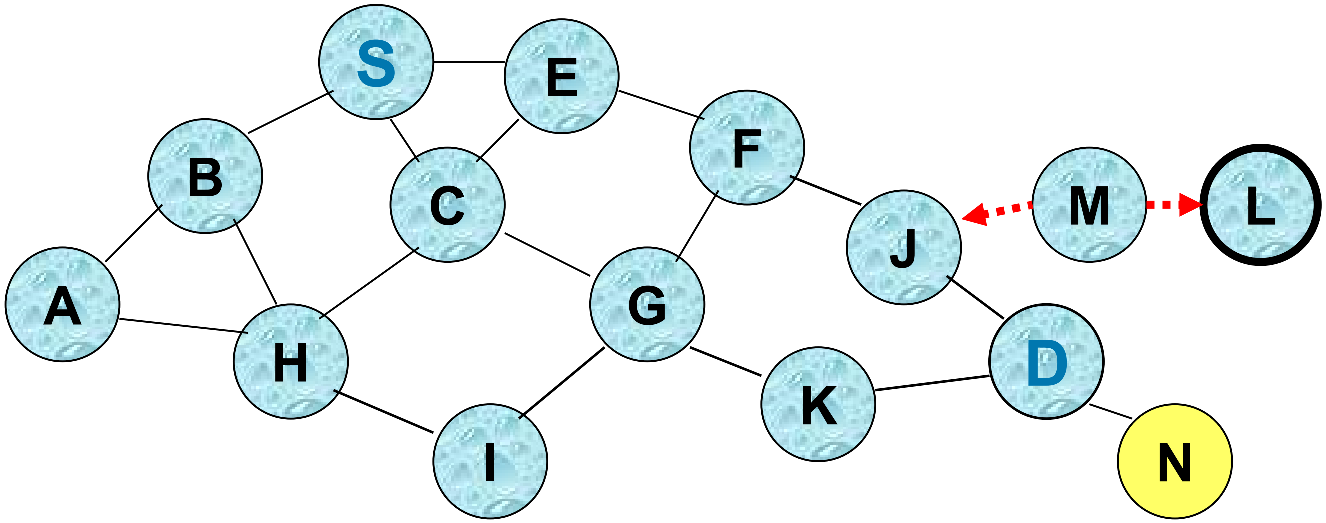
H receives packet P from two neighbors:
→ possible collision and message loss

Flooding-based Routing: Example



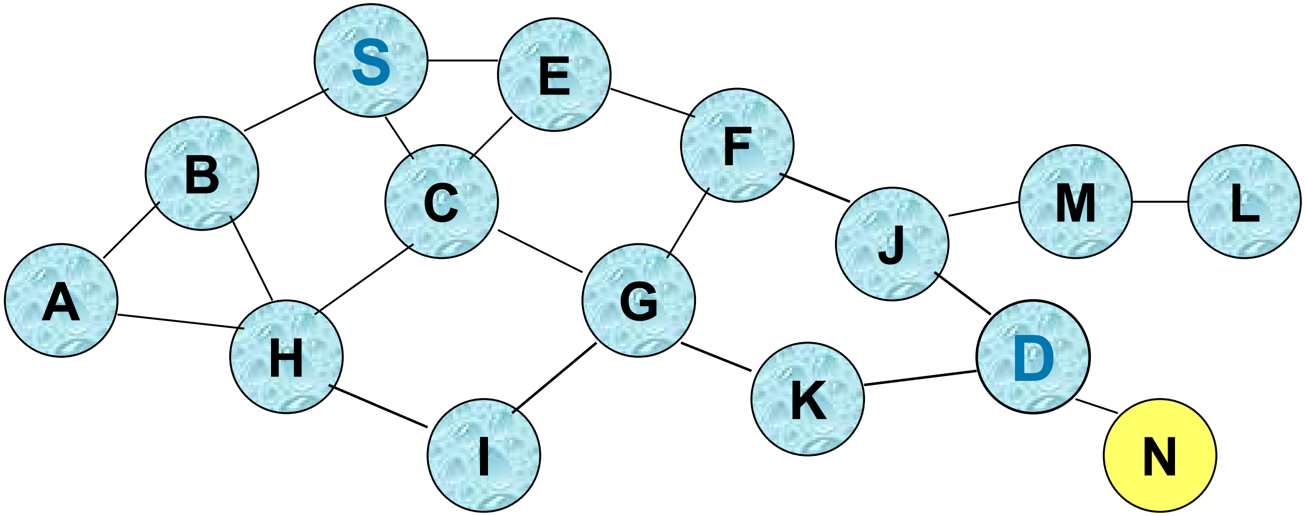
C receives packet P from G and H, but does not forward it again, because C has already forwarded packet P once

Flooding-based Routing: Example



Destination D does not forward packet P

Flooding-based Routing: Example



Flooding may deliver packets to too many nodes;
Worst case: all nodes reachable from sender may receive the packet

Task 2: Dynamic Source Routing

- Implementation of **Dynamic Source Routing (DSR)**
  - On application layer in Java using broadcast packets
  - Extend your existing code from Task 1 with route discovery mechanism control messages (Route Requests, Route Reply)

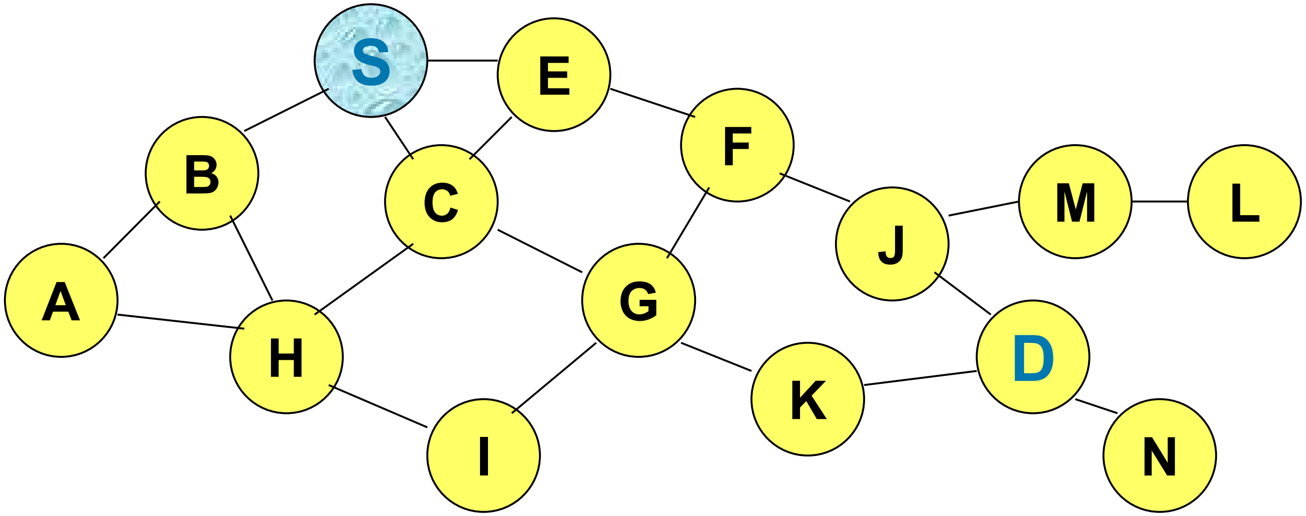
- Basic characteristic of DSR
  - Reactive and topology-based routing protocol
  - Control messages are used to discover directed routes on which data packets are sent from source to destination
  - Reduced overhead for data transfer
    - However: route discovery introduces extra message overhead

Route Discovery in DSR: Route Request

- **Protocol** (for sending RREQ)
  - Source node S floods Route Request (RREQ).
    - For flooding, the basic algorithm described above is used.
  - Each node appends own identifier when forwarding RREQ.

- **Consequently**,
  - if there exists a path from S to D, D will receive at least one RREQ message.
  - each received RREQ includes a list of identifiers defining a path from S to D.

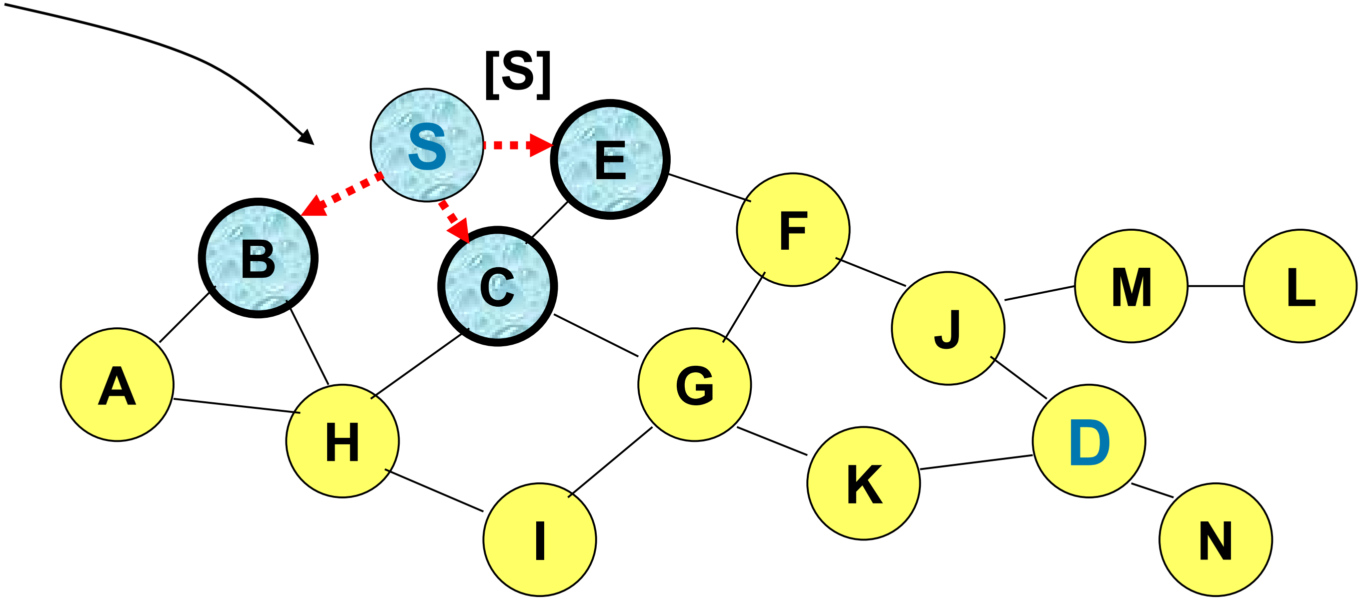
Route Discovery in DSR: Example



● Represents a node that has received RREQ for D from S

Route Discovery in DSR: Example

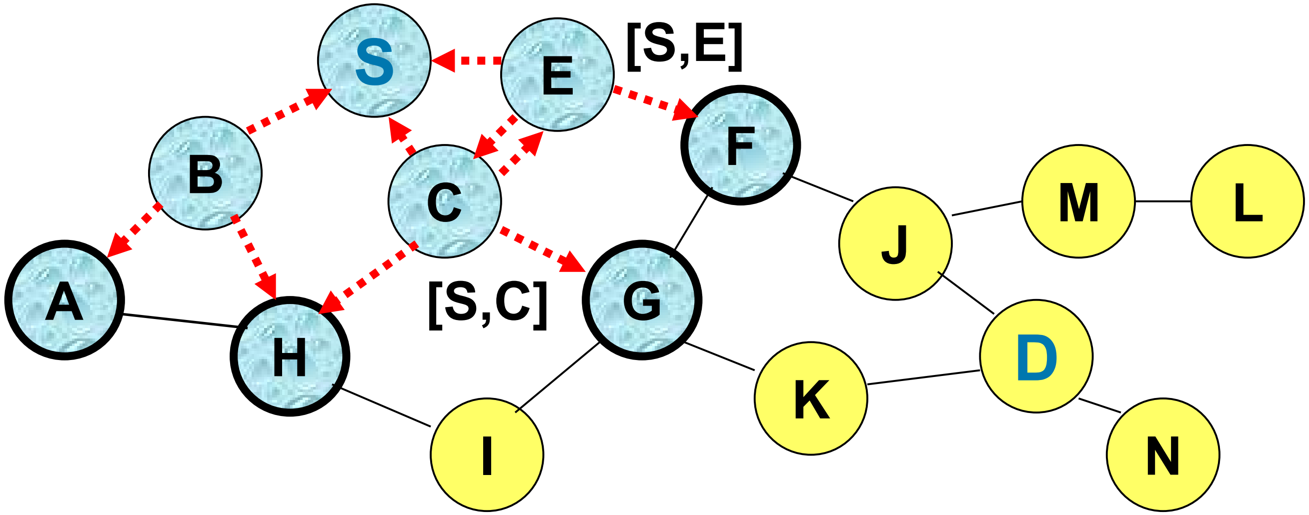
Broadcast transmission



⇢ Represents transmission of RREQ

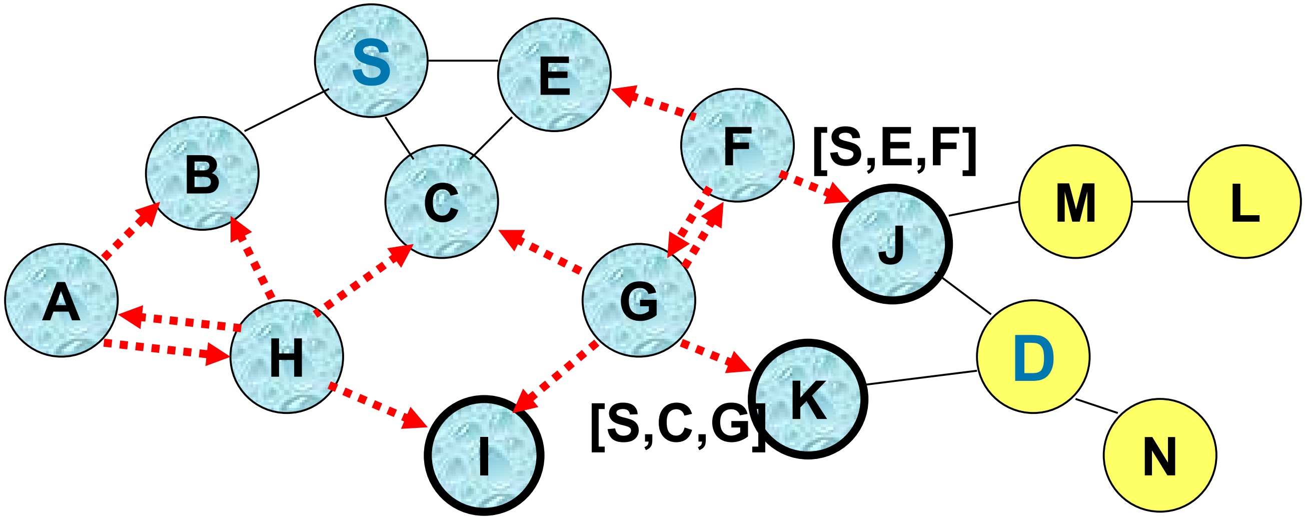
[X,Y] Represents list of identifiers appended to RREQ

Route Discovery in DSR: Example



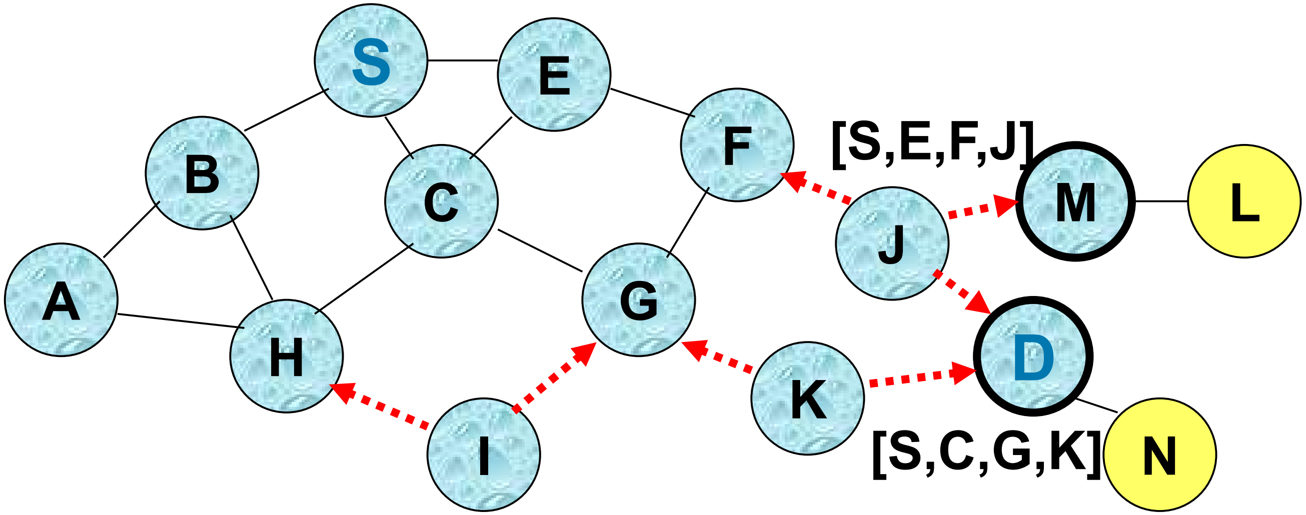
H receives packet RREQ from two neighbors: Potential for collision

Route Discovery in DSR: Example



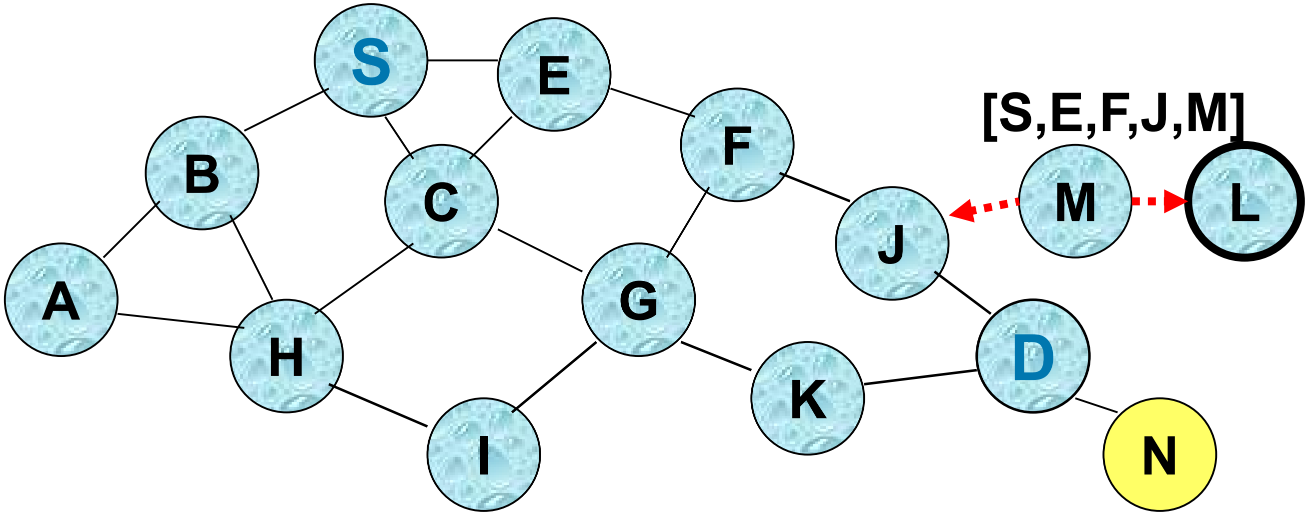
C receives RREQ from G and H, but does not forward it again, because it already forwarded RREQ once

Route Discovery in DSR: Example



J and K both broadcast RREQ to D
if J and K are hidden from each other, transmissions may collide

Route Discovery in DSR: Example



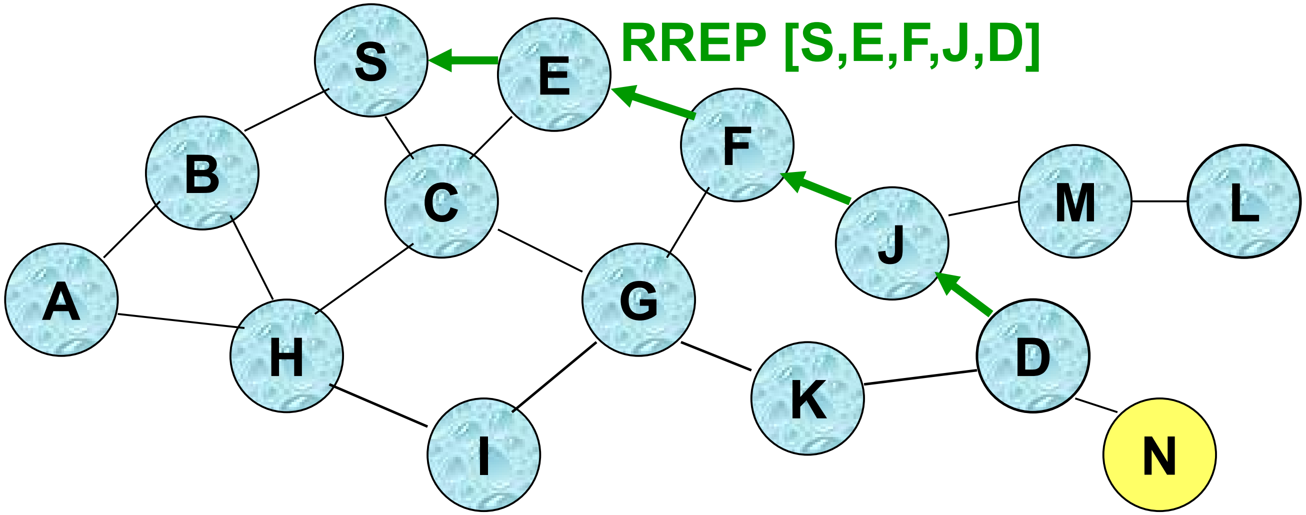
D does not forward RREQ, because it is the intended target of the route discovery

Route Discovery in DSR: Route Reply

- **Protocol (continued):**
  - Destination D, on receiving first RREQ, sends a Route Reply (RREP)
  - RREP includes the route from S to D on which RREQ was received by D
  - RREP is sent on the route obtained by reversing the route appended to received RREQ

- **Consequently**,
  - if the path included in RREP still exists, S will receive the RREP message
  - S can use the path information included in received RREP to (source) route data packets

Route Reply in DSR: Example

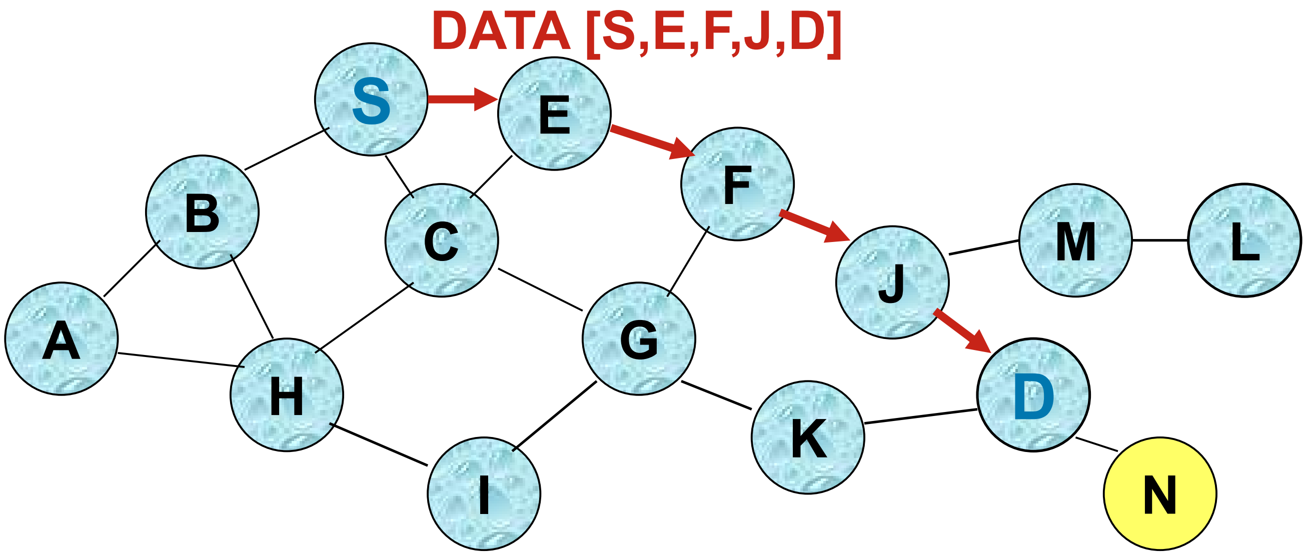


← Represents RREP control message

Data Transfer in DSR

- **Protocol:**
  - Node S, on receiving RREP, *caches* the route included in RREP
  - When node S sends a Data packet to D, the entire route is included in the packet header ➔ **source routing**
  - Intermediate nodes use the *source route* included in a Data packet to determine the next-hop node

Data Transfer in DSR: Example



Note: Data Packet header size grows with route length

DSR: Optimizations

Possible optimization to improve DSR protocol Simple routing protocol

(for a detailed explanation see lecture)

- Route caching
  - Nodes may proactively cache the routes they learn
  - Routes also contain information about subroutes
- Route Maintenance
  - Broken routes are repaired when forwarding of data fails

➡ You must implement the **basic routing protocol**

- Optimizations may be implemented voluntarily

Summary

Task 1

- Implement flooding
- **Discover** all nodes in the network
- Send message to all nodes
- **Draw graph** of network **including latency** between each node

Task 2

- Implement DSR
- Pick one host and send messages from this host to all others
- How long does route discovery need?

How to implement Flooding

- Use UDP for sending broadcast messages

```
DatagramSocket sock = new DatagramSocket();
sock.setBroadcast(true);
DatagramPacket packet = new DatagramPacket(
        bcast_msg, bcast_msg.length,
        InetAddress.getByName(“192.168.24.255”),
        5000 + team_number);
sock.send(packet);
```

- Receive broadcasts

```
DatagramSocket sock = new DatagramSocket(5000 + team_number);
DatagramPacket packet = new DatagramPacket(buf, buf.length);
while(true) {
        sock.receive(packet);
}
```

Access to the IPVS Mesh Network

Machines:

- 129.69.210.168, 129.69.210.175, 129.69.210.177, 129.69.210.152, 129.69.210.162
- **User authentication:** Username, Password as handed out in first assignment
- In computer-science network; use **marvin** as proxy

Software & Compiler:

- **JDK, Python, GCC** installed
- Mail us if you want to use other software

Important: use wifi for broadcast (wlanX); not wired networks (ethX)

- IP-Addresses: 192.168.210.168, 192.168.210.175, 192.168.210.177, 192.168.210.152, 192.168.210.162
- Only use **ports**: (5000 + x, where x is the team number)

Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Organization

Submission & Next Meeting

- Post your questions on the forum

- You have **2 weeks time** to work on this assignment until the final date of submission!
  - Demonstration of your results is scheduled for **July 4th** 2018

- **Submit via Ilias**
  - **Source code** of you evaluation results
  - Group submission!

Questions?