Routing in Mobile Ad-Hoc Networks

Organization

- Introduction to Mobile Ad hoc networks (MANETs)
- Routing in MANETs
- Virtual Backbone Routing
- Kelpi: Algorithm and implementation
- Conclusions

Towards MANETs

Networking wireless hosts:

- Cellular Networks
  - Infrastructure dependent
  - High setup costs
  - Large setup time
  - Reliable

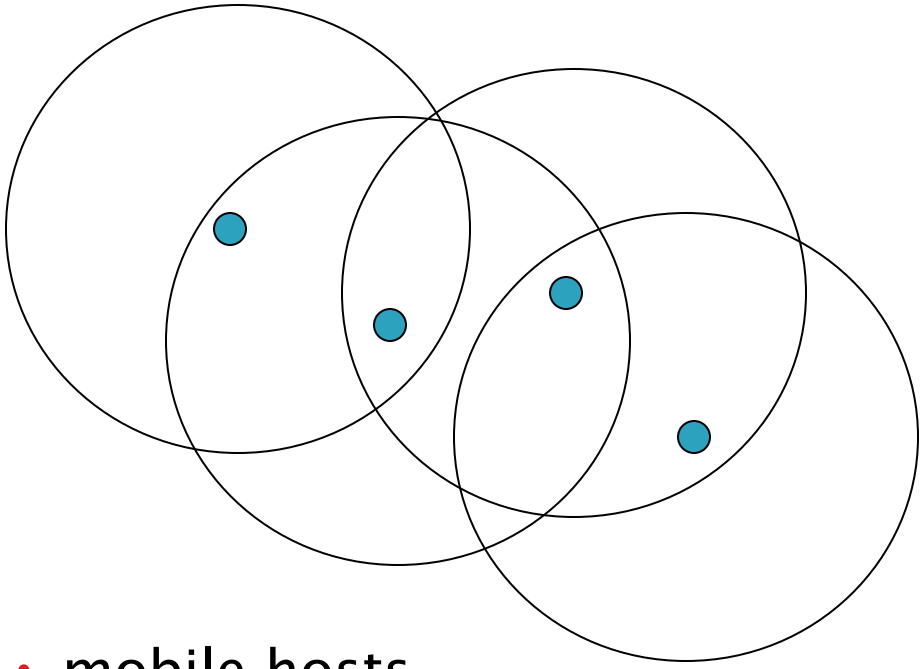
Towards MANETs

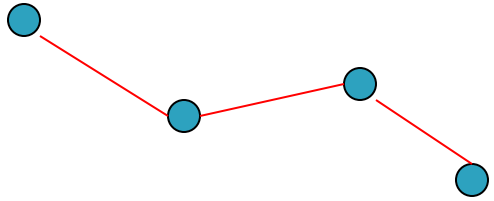
Some motivating applications:

- Casual conferencing
  - low set-up time, cost preferred
- Battlefield operations/disaster relief
  - infrastructure unavailable
- Personal area networking
  - devices around the home/office

Cellular networks are not preferred.

Mobile Ad hoc Networks



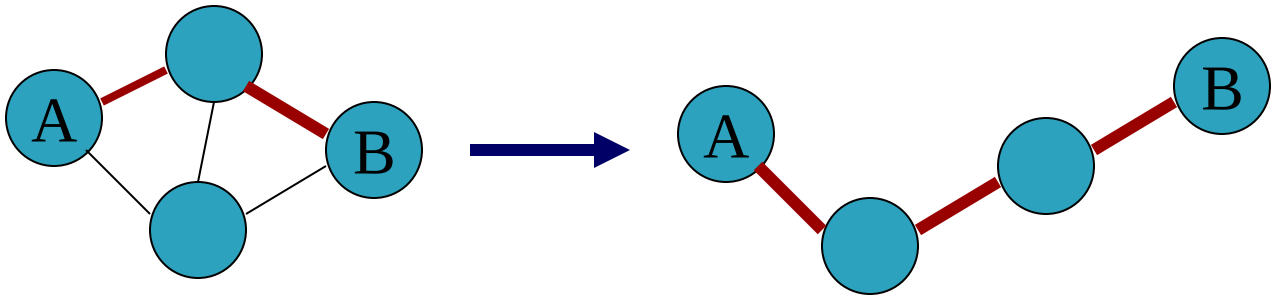


- mobile hosts
- multi-hop routes between nodes
- may not use infrastructure

Source: it644 course material
Prof. Sridhar Iyer

Characteristics of MANETs

- Dynamic topology
  - links formed and broken with mobility



- Possibly uni-directional links
- Constrained resources
  - battery power
  - wireless transmitter range
- Network partitions

Source: it644 course material
Prof. Sridhar Iyer

Routing in MANETs

To find and maintain routes between nodes in a dynamic topology with possibly uni-directional links, using minimum resources.

Dynamic Source Routing (DSR)

- Routing is through source routing
  - complete path with each packet
- Route discovery
  - flooding RREQ till a node replies
- Route maintainance
  - explicit link breakage notification

Mobility of a node can break routes passing through it.

Destination Sequenced Distance Vector (DSDV)

- Modified Distance Vector protocol
  - periodic DV updates
- High frequency of DV updates
  - topology is dynamic
- Does not scale well
  - size of DV updates increase
  - high routing overheads

Observations

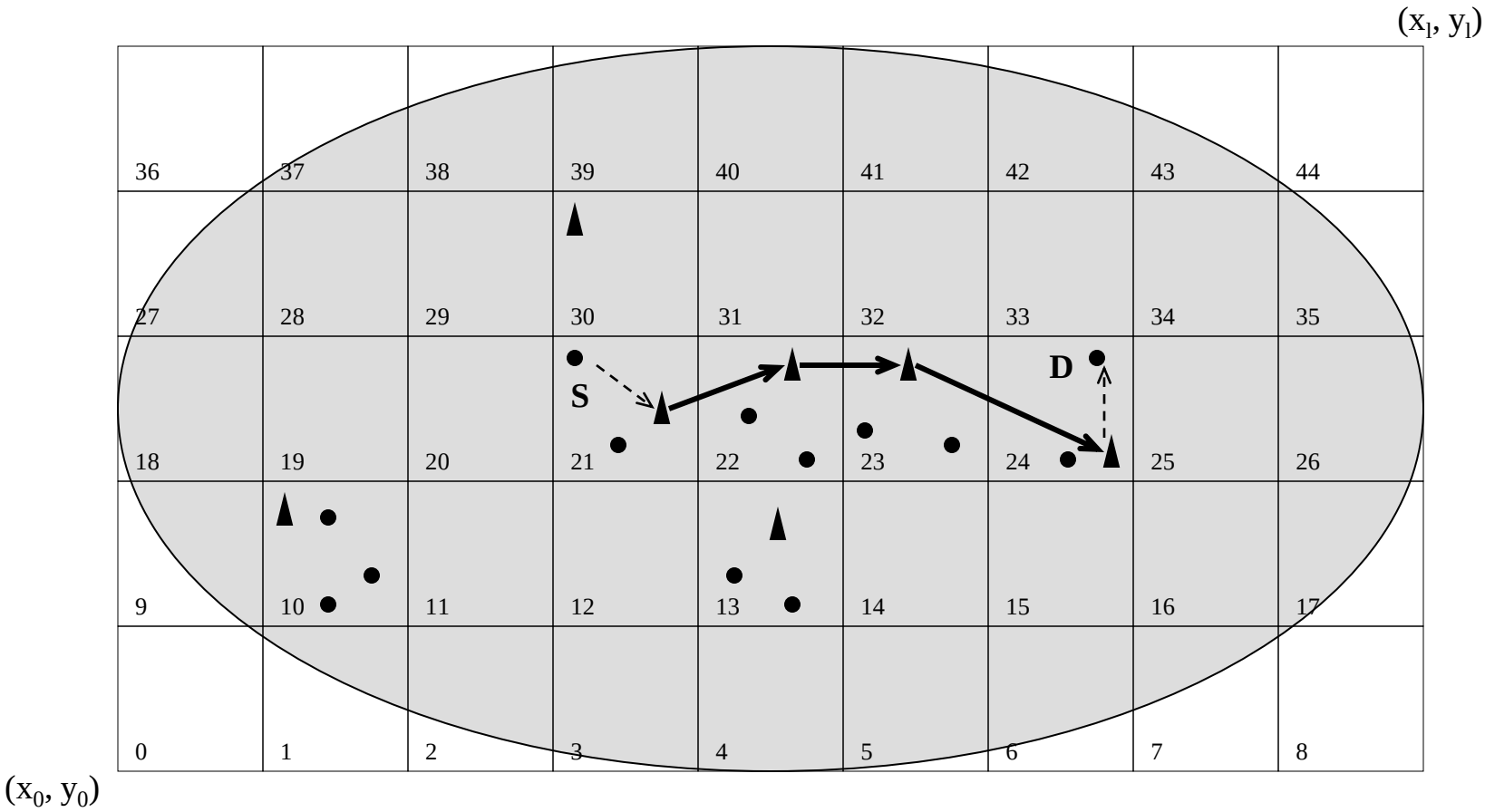
- Most ad hoc routing protocols are combinations/variations of DSR/DSDV
- Mobility in DSR causes short-lived routes
- DSDV is not scalable

The Dynamic Virtual Backbone

The dynamic virtual backbone is a concept wherein a set of relatively stable routes are formed despite nodes being mobile.

- a possible way is to abstract mobility through aggregation
  - a dynamic group of nodes by preventing some information from moving out of the group, keeps mobility transparent to the rest of the network.

Virtual Backbone in Kelpi



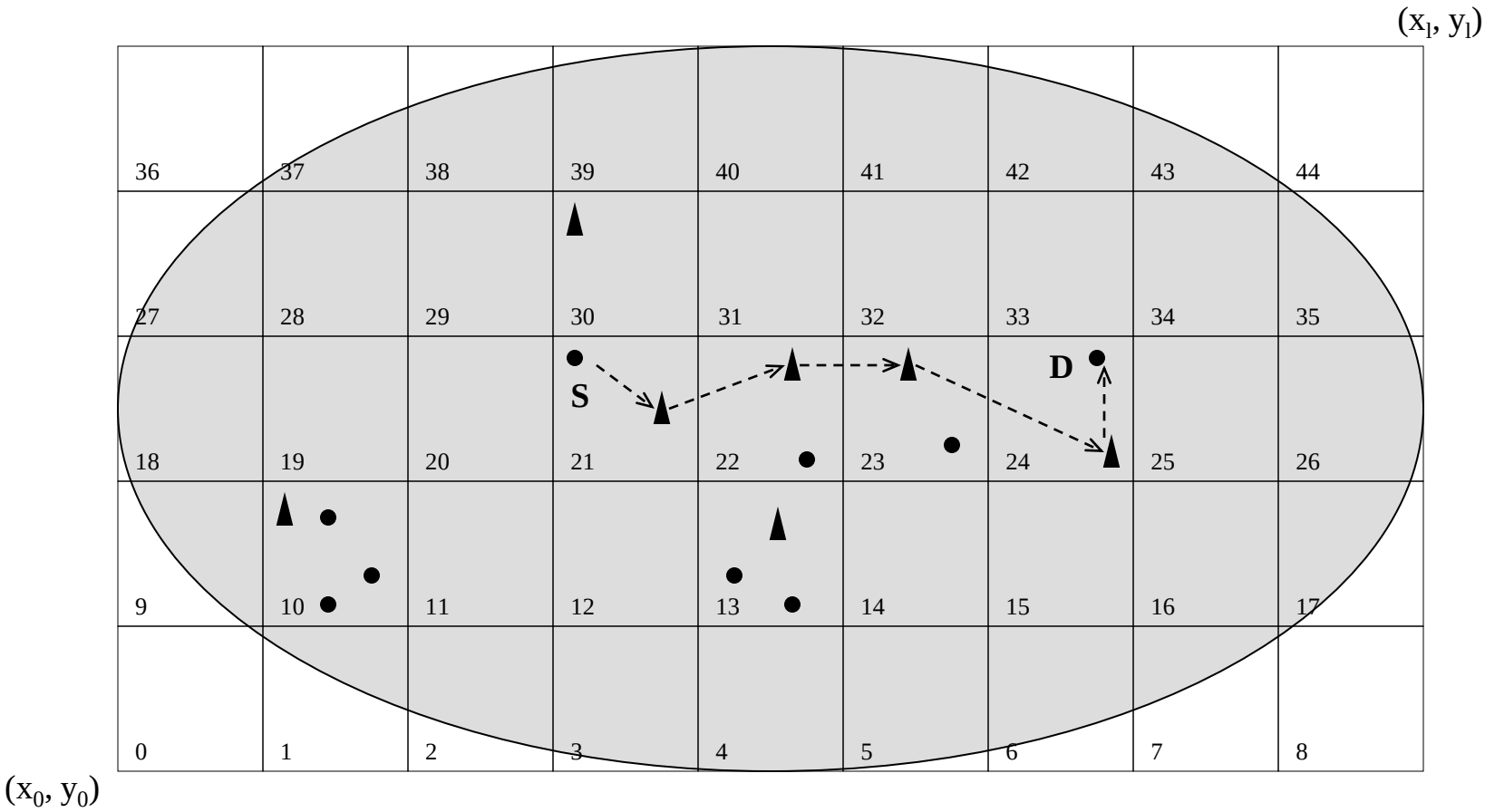
Kelpi

- Kelpi: a MANET routing algorithm based on the concept of Virtual Backbone Routing (VBR).
- Assumptions:
  - nodes equipped with positioning system, say a GPS receiver
  - nodes capable of multi-level transmission
  - mobility scenario
    - upto vehicular speeds of mobility
    - area of a few kilometres
    - fairly dense network
    - typical battlefield/disaster relief scenario

Routing in Kelpi

- Area of operation divided into square geographical cells
- In each cell one node is a router
- Inter-cell communication is through routers
  - Routers transmit at a higher transmission power
- Nodes communicate through their cell routers

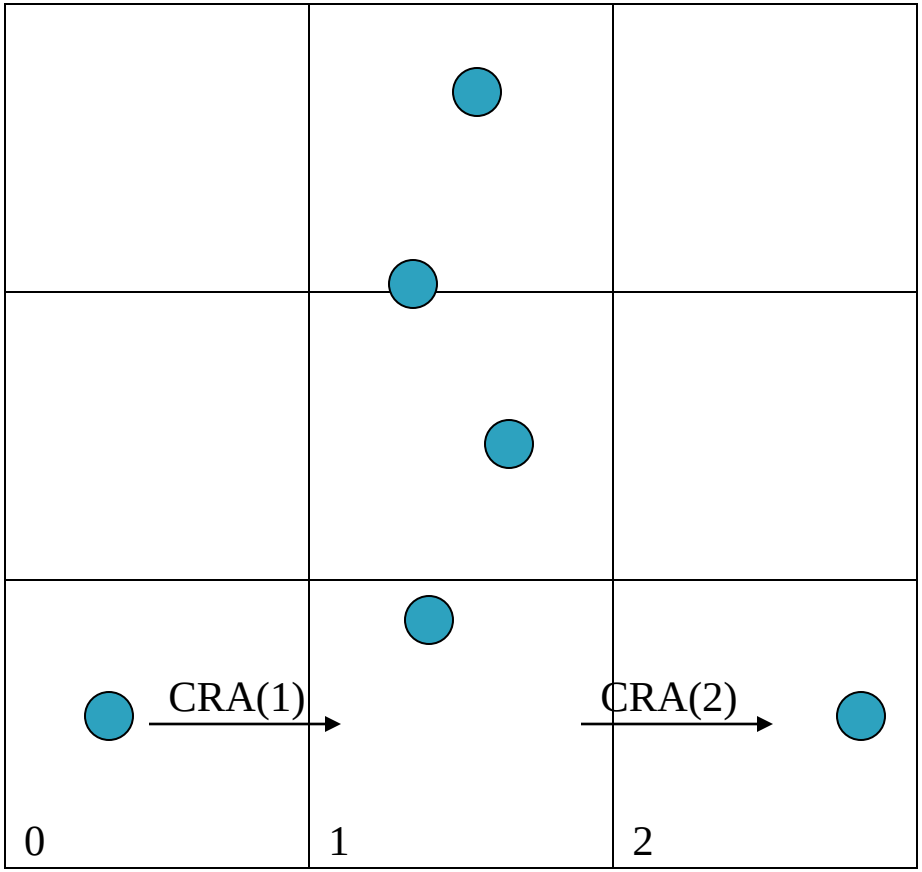
Routing in Kelpi

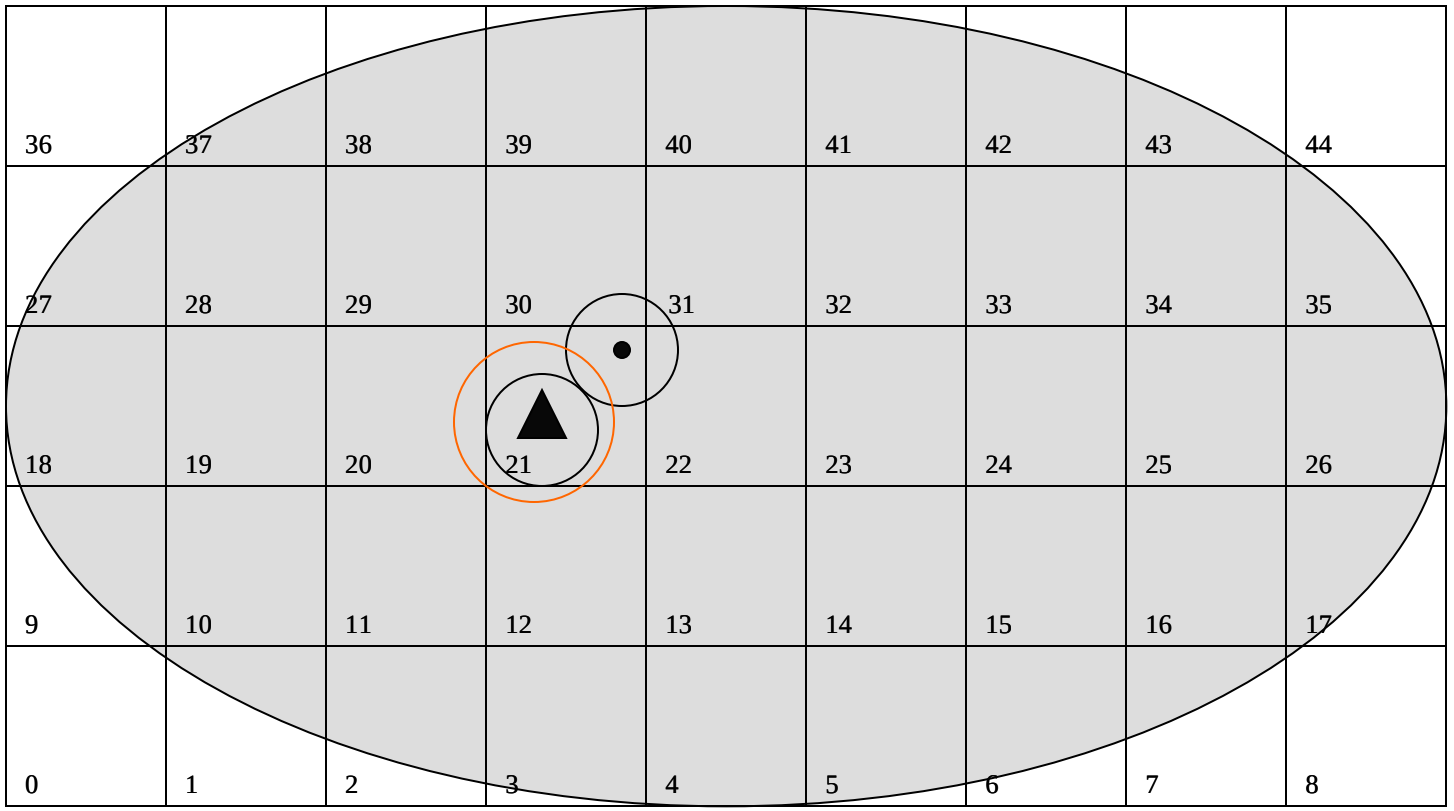


VBR in Kelpi

- Nodes aggregated by position
  - geographically defined cells
- Each group has a router
  - any node can be a router
  - router responds to a Cell Router Address (CRA)
  - before moving cells a router hands off routing information

Use of Cell Router Address and cells to implement VBR in Kelpi





Initialization:

1.Area of operation is known

2..Initialization parameters: bounding co-ordinates and maxTxPower.

Node comes on:

1.Node calculates grid

2.Node sends HI

3. Does not receive reply and declares itself router of cell 21

Another node comes on:

1.Node sends HI

2 Receives reply from router

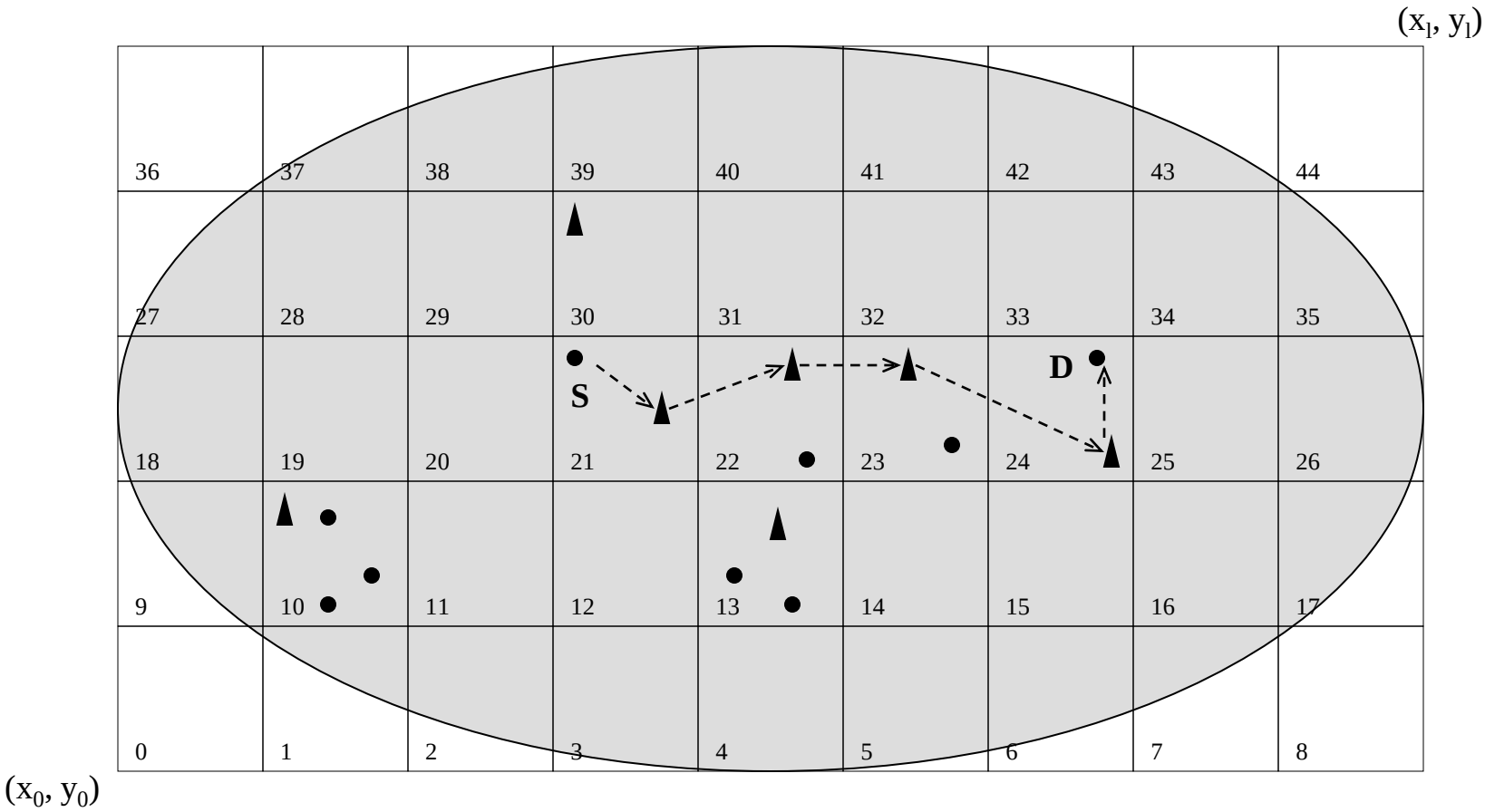
Kelpi: Router

- Data structures at router:
  - node_list, routing_table, forwarding_pointers
- The new router sends a RH (Router Here) message
  - prevents multiple routers in a cell
- starts listening on the CRA
- starts sending/receiving DV updates to/from neighbouring routers
  - <cell, distance, sequence_no>
- receives HI messages and enters sending node into node_list

Kelpi: Route Discovery

- node S wants to send to node D
  - S must know D's cell
- S discovers D's cell by sending a FIND_CELL packet to its router
- Routers flood FIND_CELL among themselves
- A router with the node in its node_list replies directly to S

Routing in Kelpi



Kelpi: Handling node mobility

- Node detects it is in a new cell
  - sends BYE to previous cell's router
  - sends HI to new cell's router
  - sends MOVED_CELL to nodes communicating with it
- Router detects it is approaching a new cell
  - initiates router handoff
    - appoints new router
    - messages: RTR_MOVE, RTR_MOVE_ACK, RTR_HANDOFF
    - sends routing_table, sequence numbers, node_list to new router
  - becomes a node

Implementation

- ns-2 network simulator used for implementing Kelpi
  - open source, used widely in MANET research
- critical modifications to ns-2
  - packet headers
  - physical layer code for multi-powered transmitter
  - introduction of new routing agent: Kelpi

Excerpt from events.cc

```
void KelpiAgent::node_receives_packet(Packet* p)
{
            struct hdr_cmn* ch = HDR_CMN(p);
            struct hdr_ip* iph = HDR_IP(p);

            int src_ip = iph->saddr();
            int dst_ip = iph->daddr();
            double now = Scheduler::instance().clock();

            // if this node originates the packet

            if(src_ip == node_address && ch->num_forwards() == 0)
        {
                        printf("ch size: %d ", ch->size());

                        ch->size() += IP_HDR_LEN;
                        printf("%d \n", ch->size());
                        iph->ttl_ = 32;         // change to num. cells in diagonal?
         }
.
.
.
                                    if ((node_cache[dst_ip] != NULL) && (node_cache[dst_ip]->time_last_accessed >
0.1) && ((now - node_cache[dst_ip]->time_last_accessed) < CACHE_STALE))
                                    {
                                                if (node_cache[dst_ip]->cell != current_cell)
                                                            forward_to_router(p, node_cache[dst_ip]->cell);
                                                else
                                                {
                                                            // send packet directly to node


                                                            ch->next_hop_ = dst_ip;
                                                            ch->addr_type_ = NS_AF_INET;
                                                            ch->txPower = nodeTxPower;
                                                            ch->src_cell = current_cell;
                                                            ch->dst_cell = current_cell;
```

```
Scheduler &s = Scheduler::instance();
                                    printf (" Direct send to %d from %d at
%lf\n",dst_ip,src_ip, s.clock());

                                    target_->recv(p,(Handler*)0);
                                    }

                                                //update cache

                                    node_cache[dst_ip]->time_last_accessed = now;
                                    }
                                    else
                                    {
                                    // buffer the packet
                                    rtQ.enque(p);
```

Excerpts from tst.tcl

```
set val(chan)         Channel/WirelessChannel  ;# channel type
set val(prop)         Propagation/TwoRayGround ;# radio-propagation model
set val(ant)          Antenna/OmniAntenna      ;# Antenna type
set val(ll)           LL                       ;# Link layer type
set val(ifq)          Queue/DropTail/PriQueue  ;# Interface queue type
set val(ifqlen)       50                       ;# max packet in ifq
set val(netif)        Phy/WirelessPhy          ;# network interface type
set val(mac)          Mac/802_11               ;# MAC type
set val(rp)           Kelpi                    ;# ad-hoc routing protocol
set val(nn)           3                        ;# number of mobilenodes
set val(txPower)      0.002w                          ;# txPower

set ns_ [new Simulator]
.
.
.
# Provide initial (X,Y, for now Z=0) co-ordinates for node_(0) and node_(1)
#
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
.
.
#
# Move

$ns_ at 1.0 "$node_(0) setdest 30.0 5.0 10.0"
$ns_ at 6.0 "$node_(1) setdest 25.0 25.0 1.0"


# TCP connections between node_(0) and node_(1)

set tcp [new Agent/TCP]
#$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(2) $sink
$ns_ attach-agent $node_(1) $tcp
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 2.0 "$ftp start"
$ns_ at 3.0 "$ftp stop"
```

Excerpt from wireless.tr

```
s 2.003625883 _2_ MAC  --- 7 tcp 1052 [a3 1 2 800] ------- [2:0 1:0 32 0] [0 0] 0 0
r 2.007833936 _1_ MAC  --- 7 tcp 1000 [a3 1 2 800] ------- [2:0 1:0 32 0] [0 0] 1 0
s 2.007843936 _1_ MAC  --- 0 MAC 38 [0 2 0 0]
r 2.007858936 _1_ AGT  --- 7 tcp 1000 [a3 1 2 800] ------- [2:0 1:0 32 0] [0 0] 1 0
s 2.007858936 _1_ AGT  --- 10 ack 40 [0 0 0 0] ------- [1:0 2:0 32 0] [0 0] 0 0
r 2.007858936 _1_ RTR  --- 10 ack 40 [0 0 0 0] ------- [1:0 2:0 32 0] [0 0] 0 0
s 2.007858936 _1_ RTR  --- 11 message 48 [0 0 0 0] ------- [1:255 0:255 32 0]
r 2.007995988 _2_ MAC  --- 0 MAC 38 [0 2 0 0]
s 2.008505936 _1_ MAC  --- 0 MAC 44 [2df 0 1 0]
r 2.008681983 _0_ MAC  --- 0 MAC 44 [2df 0 1 0]
s 2.008691983 _0_ MAC  --- 0 MAC 38 [23d 1 0 0]
r 2.008844030 _1_ MAC  --- 0 MAC 38 [23d 1 0 0]
s 2.008894030 _1_ MAC  --- 11 message 100 [a3 0 1 800] ------- [1:255 0:255 32 0]
r 2.009294077 _0_ MAC  --- 11 message 48 [a3 0 1 800] ------- [1:255 0:255 32 0]
s 2.009304077 _0_ MAC  --- 0 MAC 38 [0 1 0 0]
r 2.009319077 _0_ RTR  --- 11 message 48 [a3 0 1 800] ------- [1:255 0:255 32 0]
s 2.009319077 _0_ RTR  --- 11 message 48 [a3 0 1 800] ------- [0:255 -1:255 31 0]
```

Kelpi: Implementation

- following functionality has been successfully implemented
  - topology related functions
  - cell discovery
  - destination cell caching
  - packet buffering
  - packet forwarding
  - router hand-offs
- these have been validated for small test cases

Kelpi vs. other algorithms

- Advantages
  - designed to provide stable routes
  - increased throughput due to two levels of transmission
  - reduced flooding overhead
- Disadvantages
  - positioning system required
  - muliple levels of transmission preferred
  - routers may be overloaded in a dense network

Future Directions

- Remove requirement of GPS from Kelpi
- Generalize concept of Virtual Backbone Routing to other existing routing algorithms