Shikav extensions to support networking animation

Moniphal Say

Under the guidance of:

Prof. Sridhar Iyer and Prof. Abhiram Ranade

Dept. of CSE (KReSIT)

16th July, 2007

Snap shot of electronic lesson in Shikav

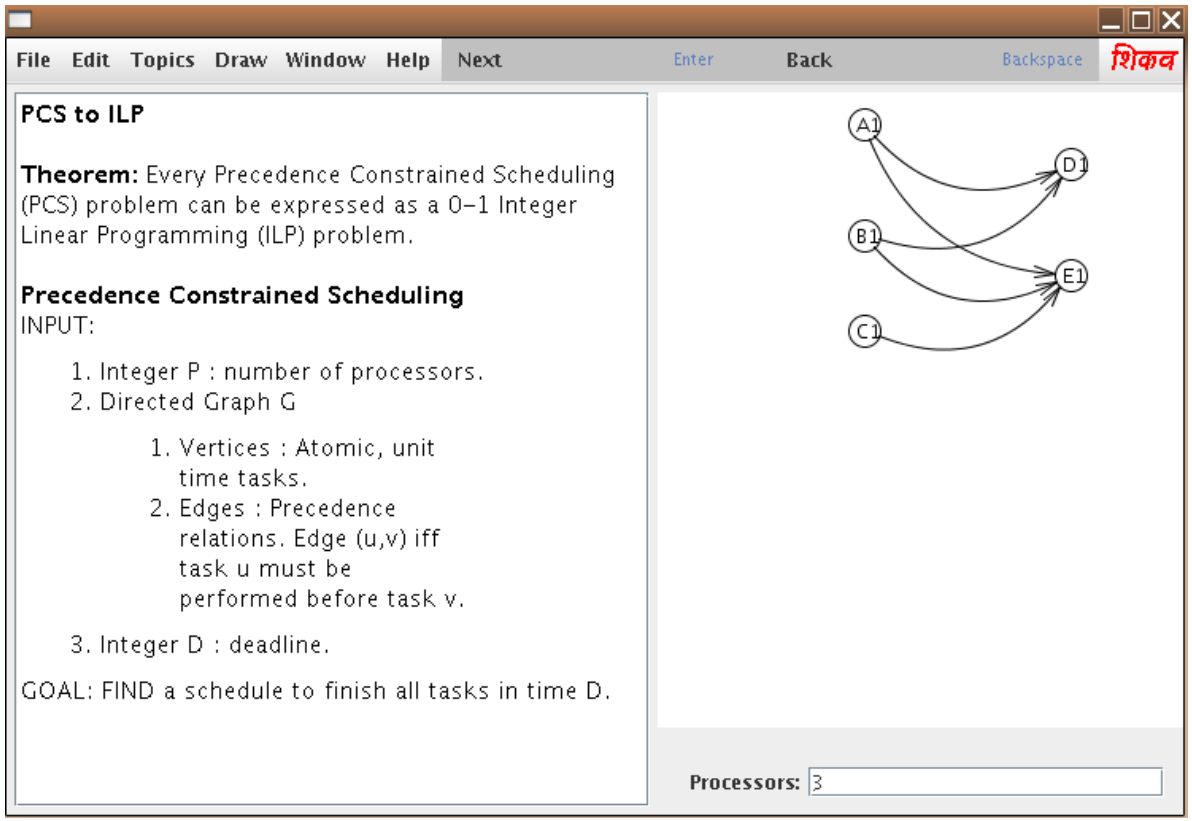


Figure: The snap shot of animation in Shikav

What is missing

Something is missing?

- Intrinsic support for animations explaining network protocols

Goal

Goal

- Enhancement of Shikav framework to support networking animation
- Define and implement a high level script language for networking
- Representation of how packets have been transferred

Out of scope

- No simulation
- No analysis of the packets

Background

Avantages of electronic lesson:

- Multimedia capability
- Interaction
- Highlight and summary
- Availability on web

Why Shikav?

- Shikav uses “reflection” technique of Java - easy to make availability of new added classes from its script language
- Shikav has stage metaphor - classroom whiteboard model

Snap shot of electronic lesson in Shikav

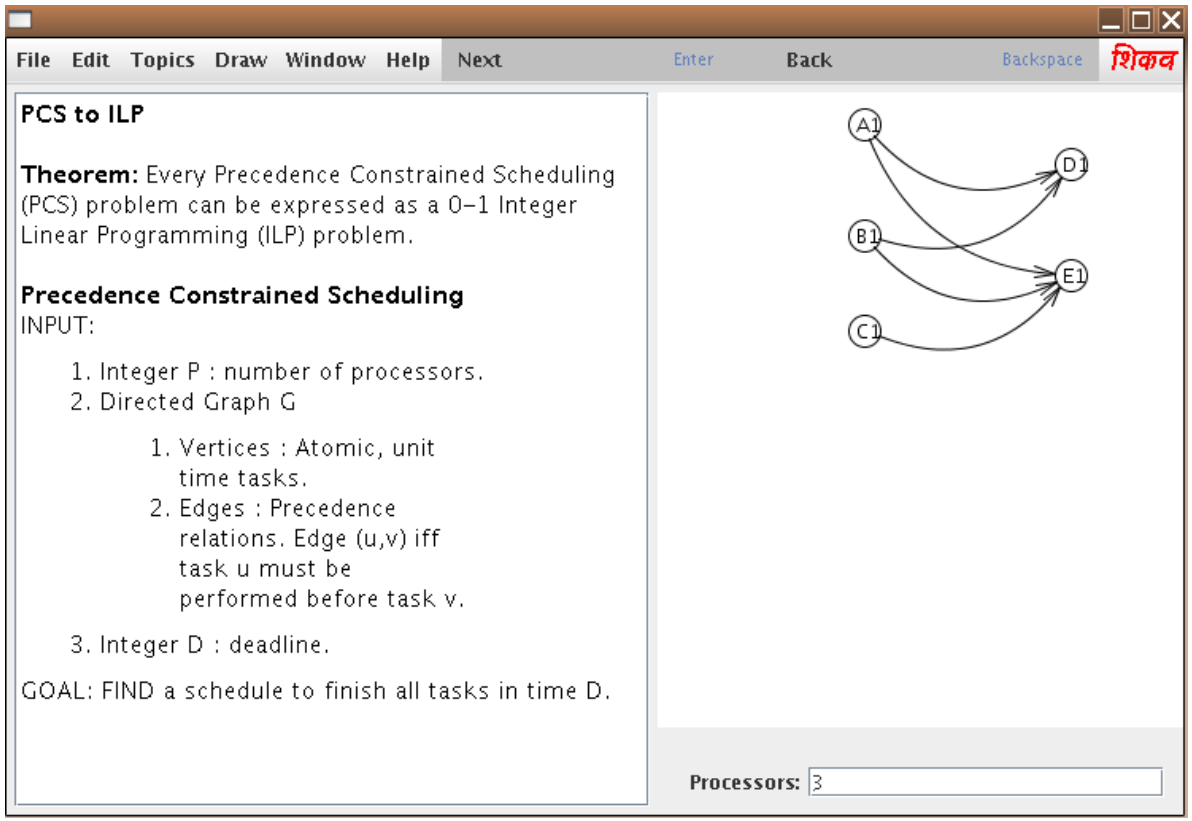


Figure: The snap shot of animation in Shikav

Overview of Shikav

- Framework for creating animations
- It has its own script language

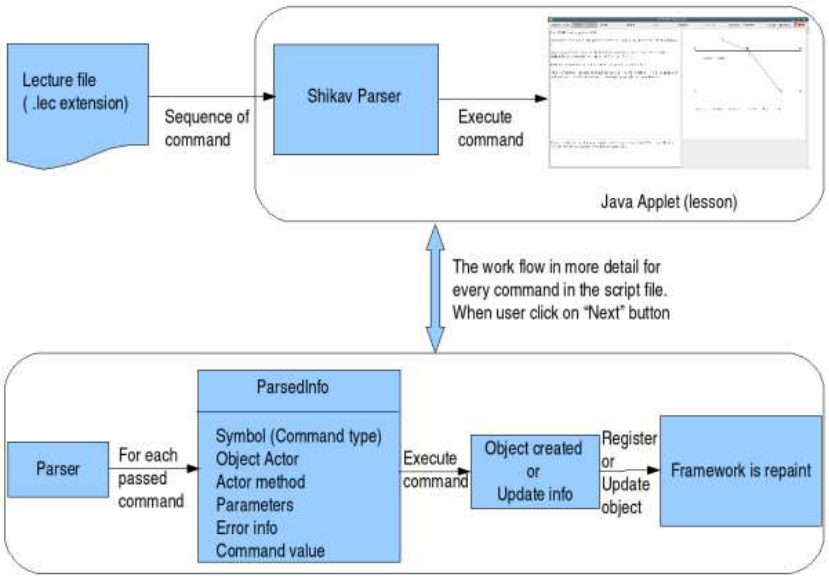


Figure: The work flow of Shikav

Existing features of Shikav

- Basic inbuilt objects (point, line, circle etc...)
- Double view editor - view lecture file while animation running
- GeoShikav - freehand drawing
- Mathematical expression - complex math expression
- Group and its operations: map, reduce, filter

Components of Shikav

Working phases of Shikav

- Compile time
- Issue time (creation of objects)
- Update time

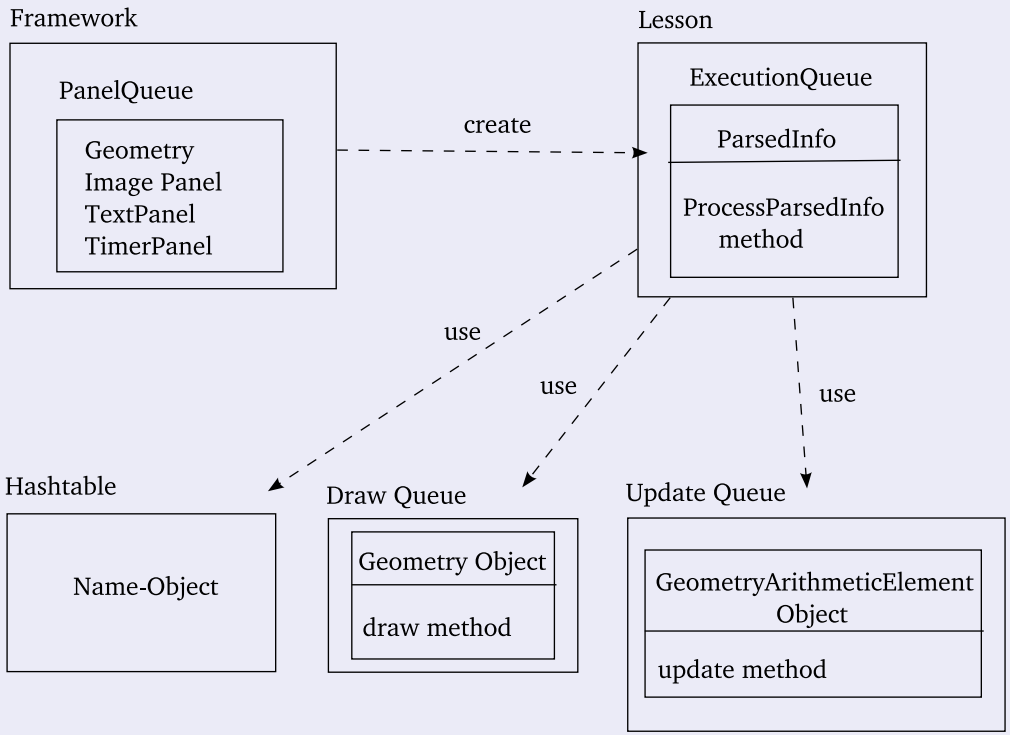


Figure: Component of Shikav

TCP 3 way handshake

Protocol steps

1) A –> B SYN
2) A <– B ACK
3) A <– B SYN
4) A –> B ACK

Step 2 and 3 can be combined, hence called “3 way handshake”

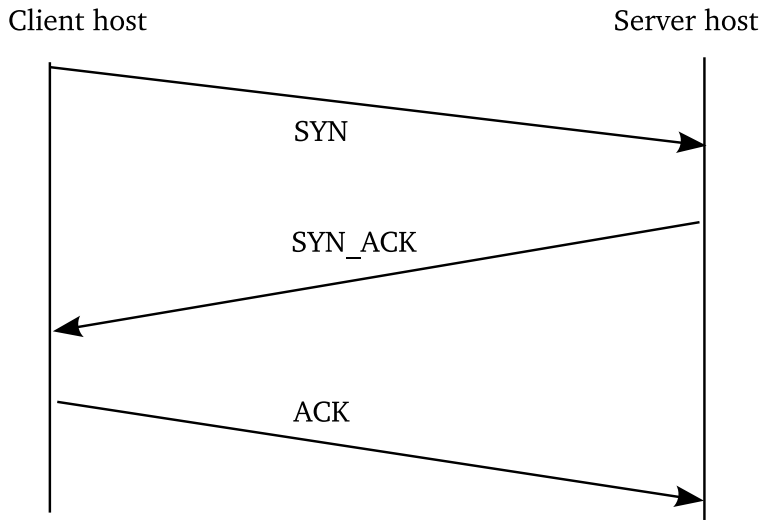


Figure: The process of TCP 3 way handshake (TCP initial connection set up)

TCP slow start

Avoid congestion in the network. Its algorithm is as follows:

set cwnd = IW (Initial Window) = 1 or 2

set ssthresh = 65535

Repeat the procedure below until cwnd <= ssthresh

send cwnd number of packet

receive ACK

cwnd = cwnd*2

Entering “congestion avoidance” phase, ”cwnd = cwnd + 1”, if time out occurs

set ssthres = cwnd/2

set cwnd = IW

Features needed are: Node, Packet, loops

WiFiRe overview

- Newly designed protocol for rural connectivity
- A system divided into sectors
- Multiple STs in each sector, EUs connect to ST

Basic steps of the protocol

- Ranging
- Registration
- Data connection

Features needed: Node, Packet, Beacon

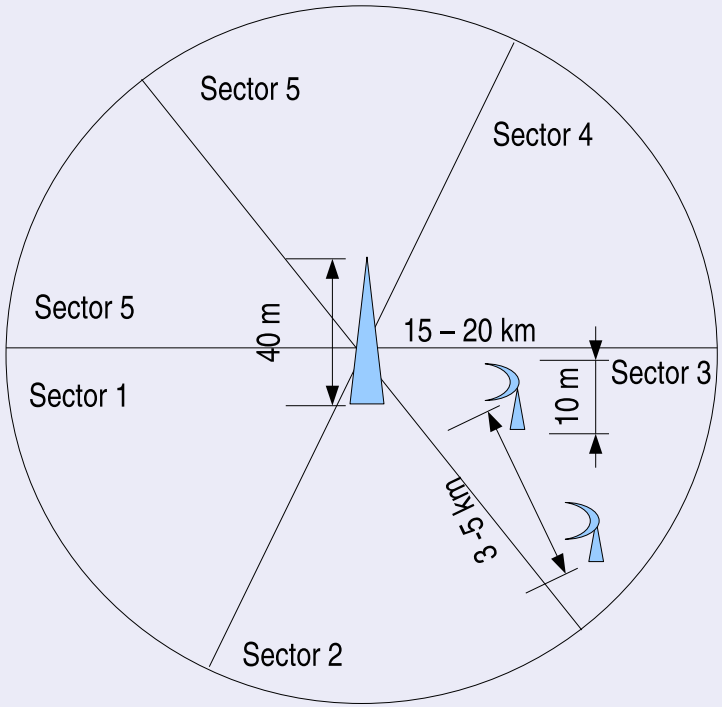


Figure: Overview of WiFiRe

What is missing? - Features required to be added

- Node: represents node in networks
- Packet: represents packets in networks
- Beacon: is required in case for protocol like WiFiRe
- loops: is required in case an author wants to create many packets or send many packets - TCP slow start
- send behavior: is required for sending packets
- receive behavior: is required for receiving packets
- If-Else conditional statement: test the content of packet receive to differentiate the response action

Class diagram - EntityNode

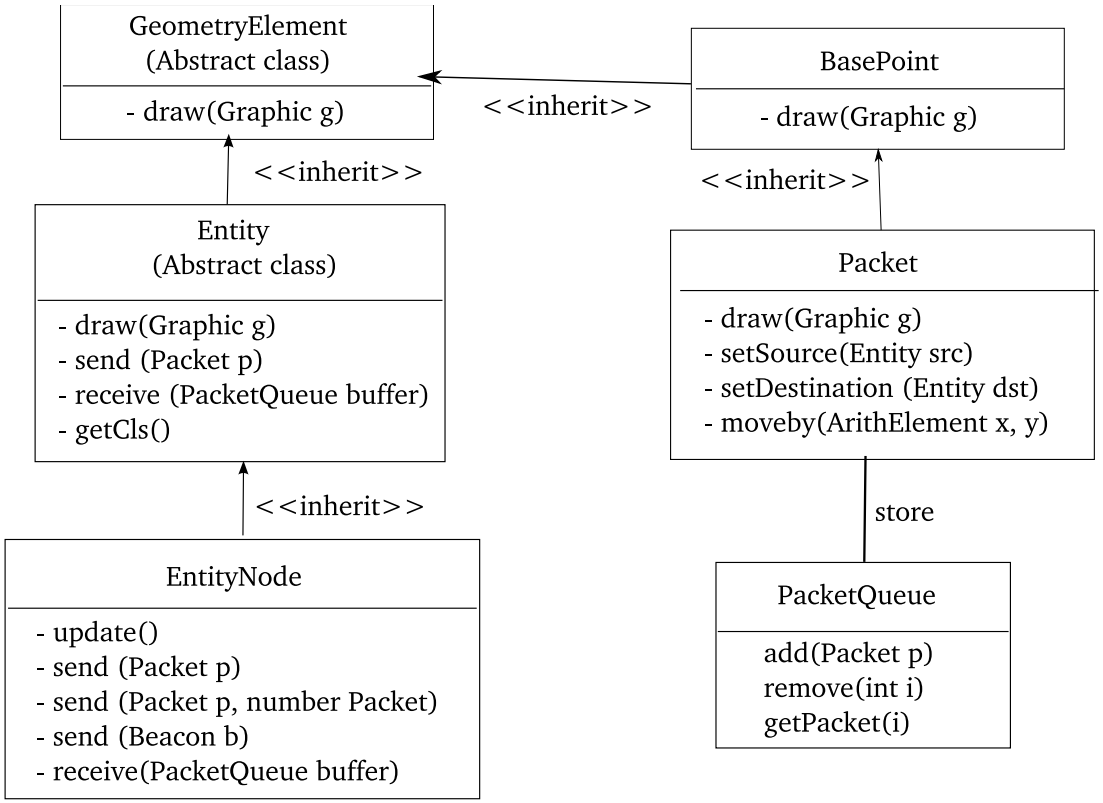


Figure: Relation between newly added classes

Class diagram

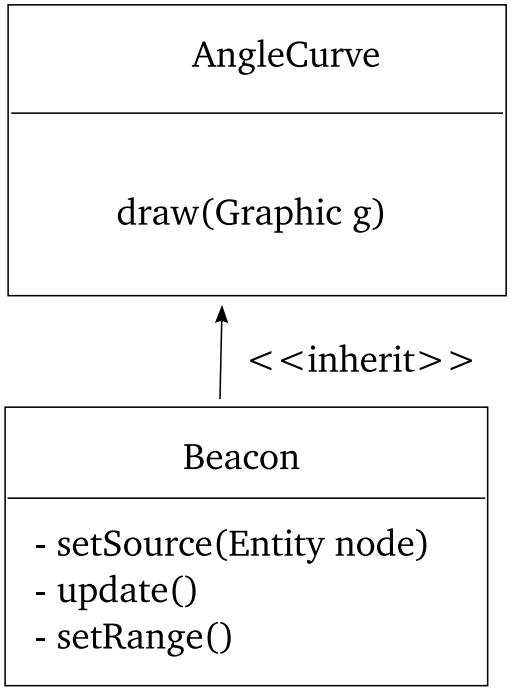


Figure: Relation between newly added classes

Shikav script enhancement

What we can do with the new added class:

- create nodes
- create beacons
- create packets
- send, receive packets
- send beacons

Network script language

A high level scrip language is defined and implemented

- for the sole purpose of creating networking animation
- to define the behavior of nodes
- facilitate the author in creating networking animation

Syntax of the script language

- **Node** *node-name*
- **Node** *node-name x y*
- **ConstructPacket**(*source, destination, content, packet-name*)
- **send**(*packet-name, source, destination*)
- **send**(*packet-name, source, destination, number-pkt-to-send*)

Network script language

Syntax of the script language

- **ConstructBeacon**(*source, content, range, angle, beacon-name*)
- **send**(*beacon-name, source*)
- **repeat** *constant variable* **end**
- **If** (*condition*) **Then** *statement* **Else** *statement* **endif** "condition = *node-name* **receive** *content*"
- **delay** (*seconds*)
- **title** *title of the animation*

Example of network script language - TCP slow start

```
title "TCP Slow Start"
node A
node B
constructPacket(A , B , "payload" , p)
send(p, A, B)
repeat 15 i
if (A receive "ACK") then
constructPacket(A , B , "payload" , p)
send(p, A, B, 2^i)
else
endif
```

Snap shot of TCP slow start lesson

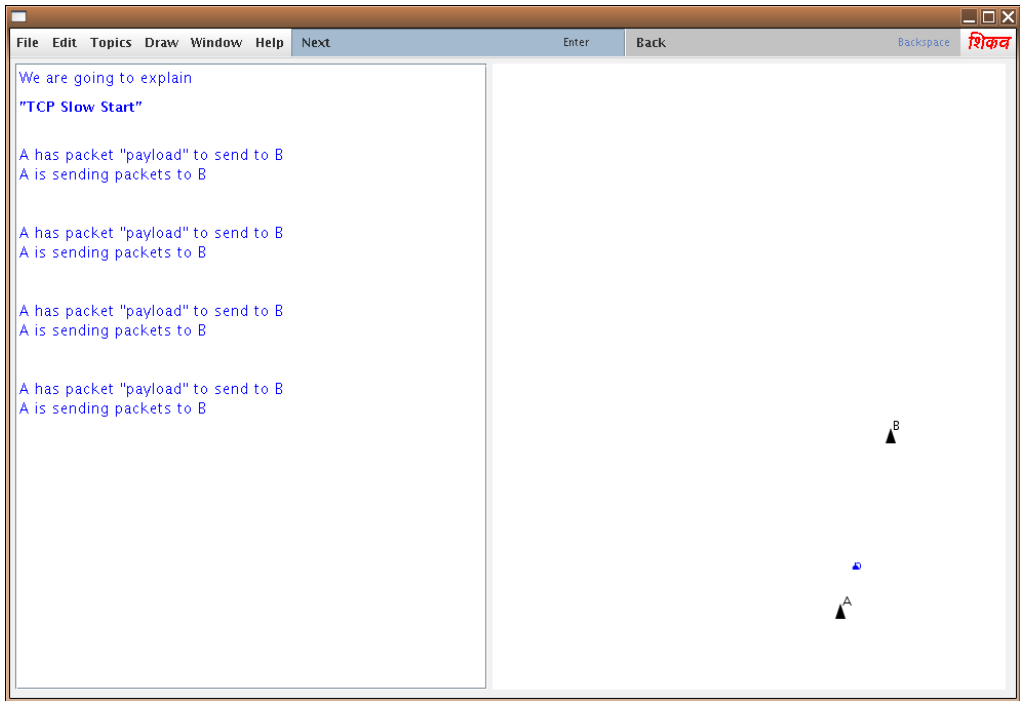


Figure: Snap shot of animation showing the working of TCP Slow Start

Example of network script language - TCP 3 way handshake

```
Title "TCP 3 way handshake"
Node HostA
Node HostB
ConstructPacket(HostA , HostB , "SYN" , p1)
send(p1, HostA, HostB)
if (HostA receive "SYN") then
ConstructPacket(HostB,HostA,"SYN_ACK",p2)
send(p2, HostB, HostA)
else delay(2)
endif
if (HostB receive "SYN_ACK") then
ConstructPacket(HostA, HostB, "ACK", p3)
send(p3, HostA, HostB)
else delay(2)
endif
```

Snap shot of TCP 3 way handshake lesson

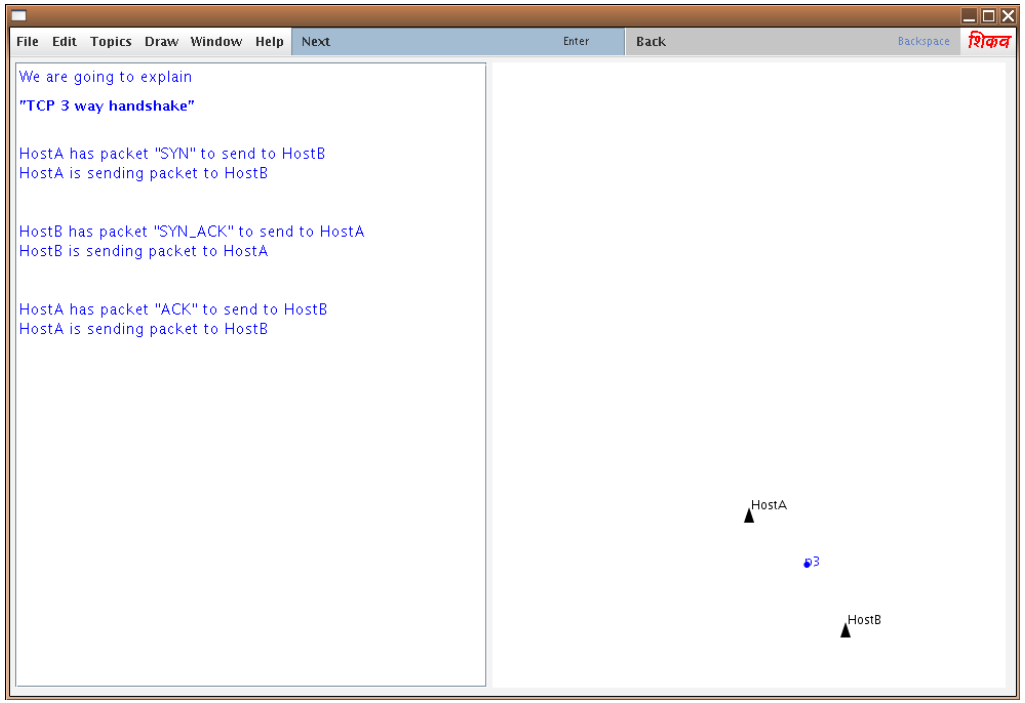


Figure: Snap shot of TCP 3 way handshake lesson

```
Title "WiFiRe"
Node BS 20 30
Node ST 150 200
Node EU 50 200
ConstructBeacon(BS, "beacon",
400, -60, b)
send(b, BS)
ConstructPacket(ST , BS , "IRRe"
, irre)
send(irre, ST, BS)
if (BS receive "irre") then

ConstructPacket(BS,ST,"IRRes",irres)
send(irres, BS, ST)
else
endif
if (ST receive "IRRes") then
ConstructPacket(ST, BS,
"RegReq", regreq)
send(regreq, ST, BS)
else
delay(1)
```

```
ConstructPacket(EU, ST, "SYN",
syn)
send(syn, EU, ST)
if (ST receive "SYN") then
ConstructPacket(ST, BS,
"DSA-Req", dsareq)
send(dsareq, ST, BS)
else
delay(1)
endif
if (BS receive "DSA-Req") then
ConstructPacket(BS,ST,"DSA-
Res",dsares)
send(dsares, BS, ST)
else
delay(1)
endif
if (ST receive "DSA-Res") then
ConstructPacket(ST, EU,
"SYN_ACK", synack)
send(synack, ST, EU)
else
```

Snap shot of WiFiRe lesson

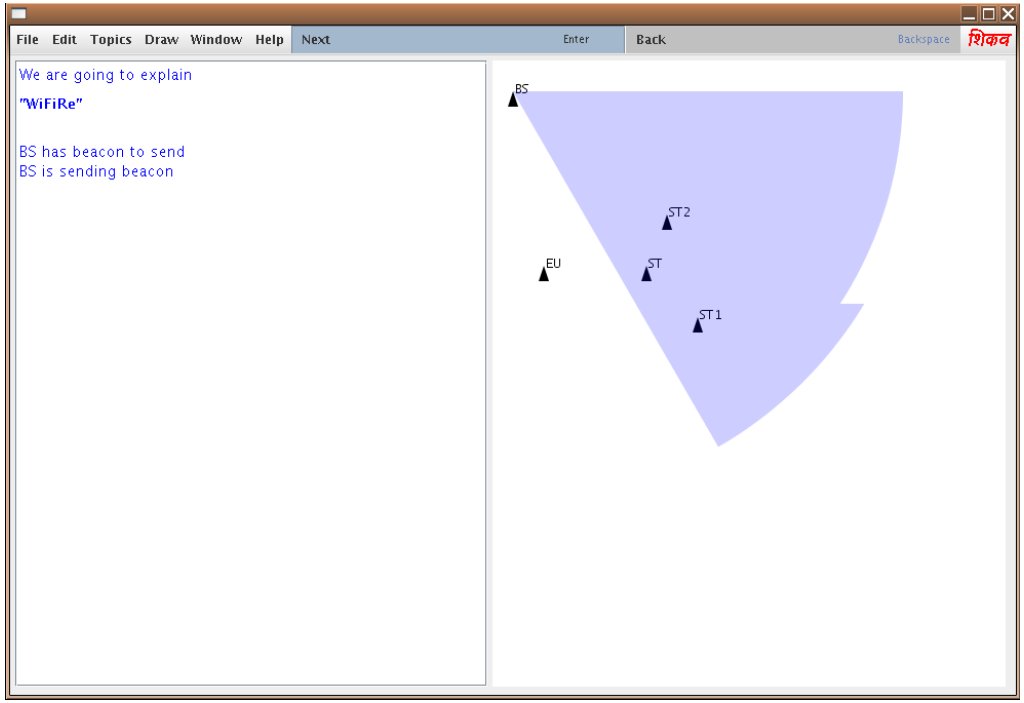
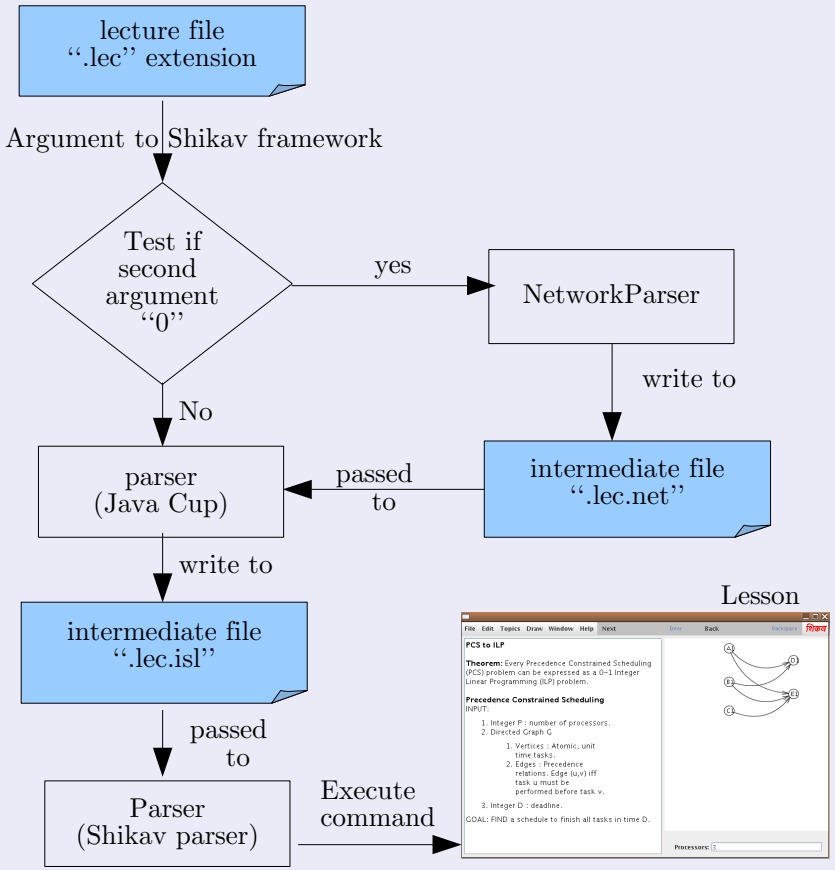


Figure: Snap shot of animation showing the working of WiFiRe protocol

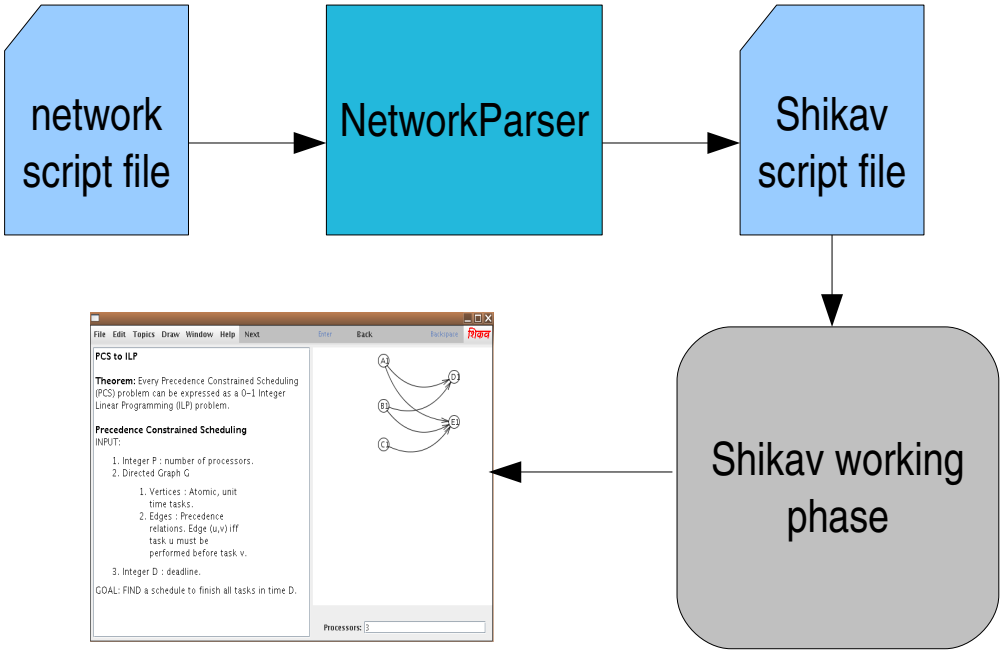
Integration into Shikav



NetworkParser - How does it work?

- Read from the input file “.lec”
- If the command starts with key word “Node, ConstructPacket, ConstructBeacon, Send, Repeat, If, Delay, Title”.
- Translates each command into corresponding Shikav script language, and write them into “.lec.net” file.
- Special case for *Repeat*, has to read a block of command till *end*, then parse as normal command
- Special case for *If*, has to read a block of ***Then statement*** and ***Else statement*** and parses them as normal command.
- Set the file type to “org” and passed it to the existing phase parser of Shikav.

Network parser



The flow of network script file

Conclusion and Future extensions

Conclusion

- Network features are added into Shikav - allows to create animation explain network protocols -> adding intrinsic support for networking animation
- Network script language is defined and implemented - facilitate the author in creating lesson explaining network protocols
- Goal archieve

Future extension

- Integration of simulation capability and the new enhancement to Shikav
- Drag/Drop node

Thank you !

Backup slides

Back up slides

Challenge

- Was not able to show the animation of packet within the “send” method of EntityNode, even “update” method is called explicitly - reason, object is not register in the Geometry panel
- Adding If-Else conditional statement - previously defined in user defined function which accept only arithmetic expression. There are too many steps in order to get into that function
- The big issue in If-Else conditional statement is during update time of Shikav, which it need to call “update” method
- Trying to understand Shikav in detail

DNS - lookup

node client

node DNS

constructPacket(client, DNS, "http://akash.it.iitb.ac.in",p)

send(p, client, DNS)

if (DNS receive "http://akash.it.iitb.ac.in") then

delay(2)

constructPacket(DNS, client, "10.129.1.2", res)

send(res, DNS, client)

else

endif

DNS - lookup

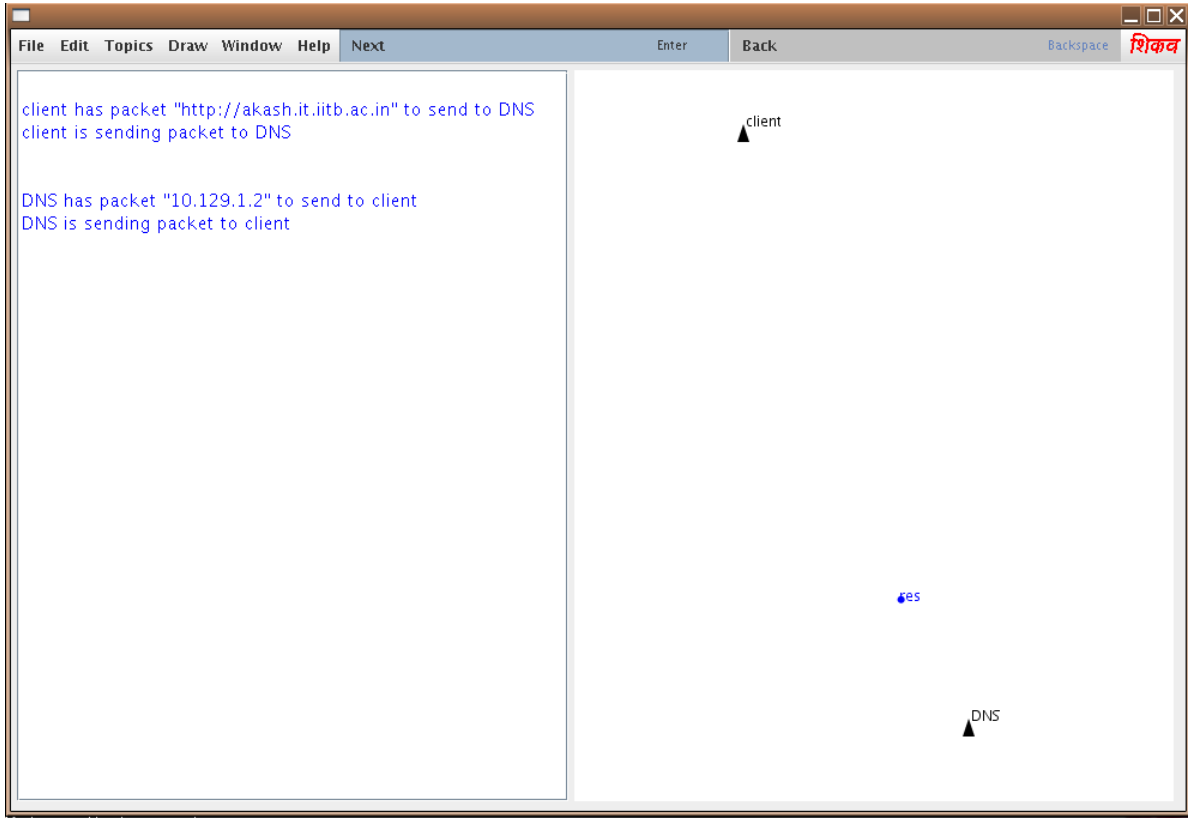


Figure: Snap shot of DNS look up

JSIM - Java Simulation

JSIM : Web-based simulation developed by John A.Miller, Andrew F. Seila and Xuewei Xiang Features

- Graphical User Interface, Generate Java applet code
- Nodes and properties related simulation in queuing network
- Distribution function, service time, etc ...

JSIM - Java Simulation

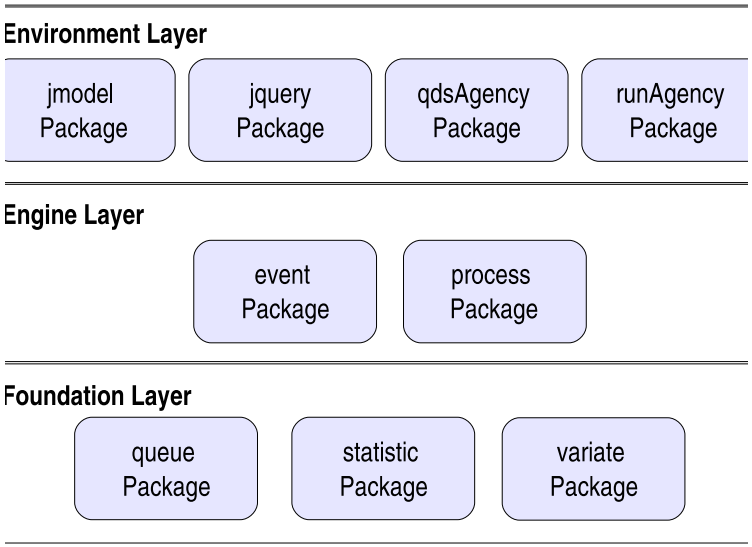


Figure: Layer of JSIM

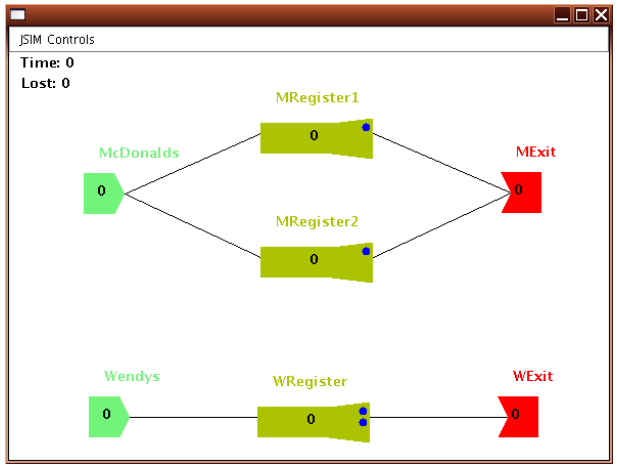


Figure: Snap shot of simulation in JSIM

WiFiRe - Ranging

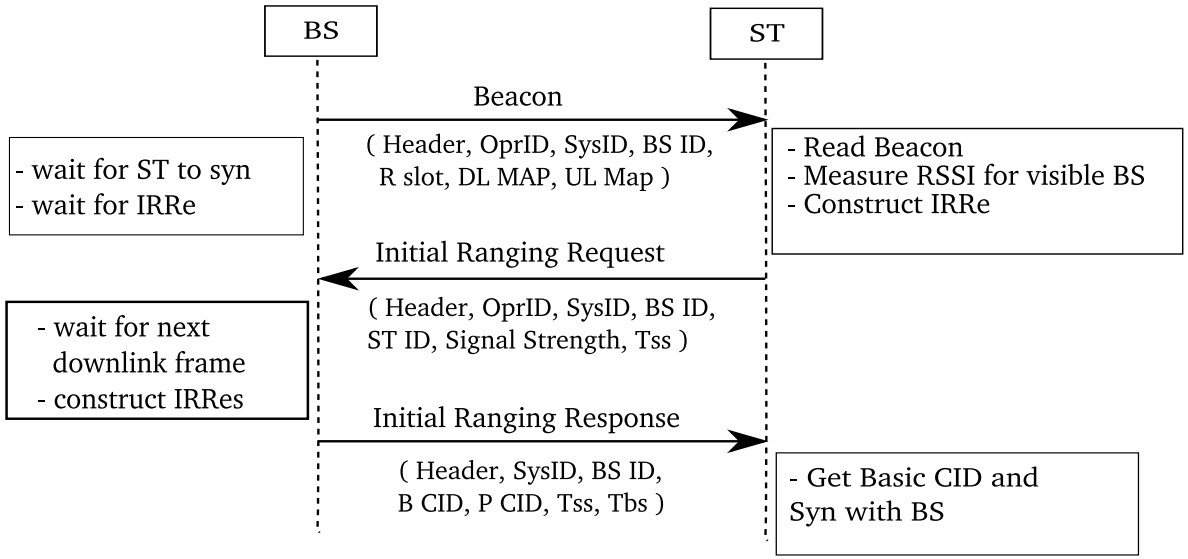


Figure: Sequence diagram of initial ranging steps

WiFiRe - Registration

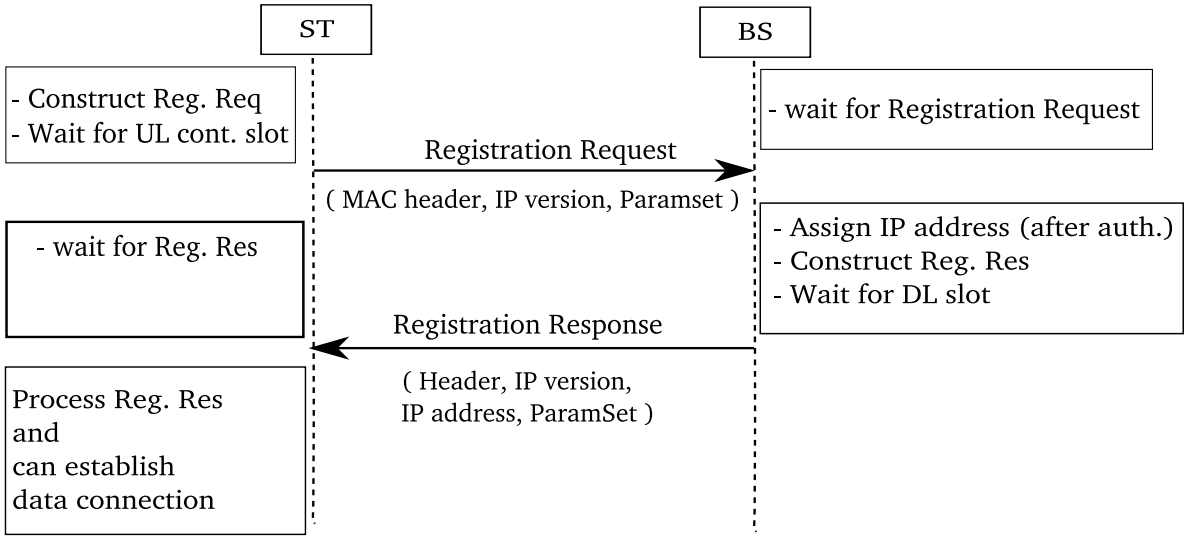


Figure: Sequence diagram of registration steps

WiFiRe - Data connection - UGS

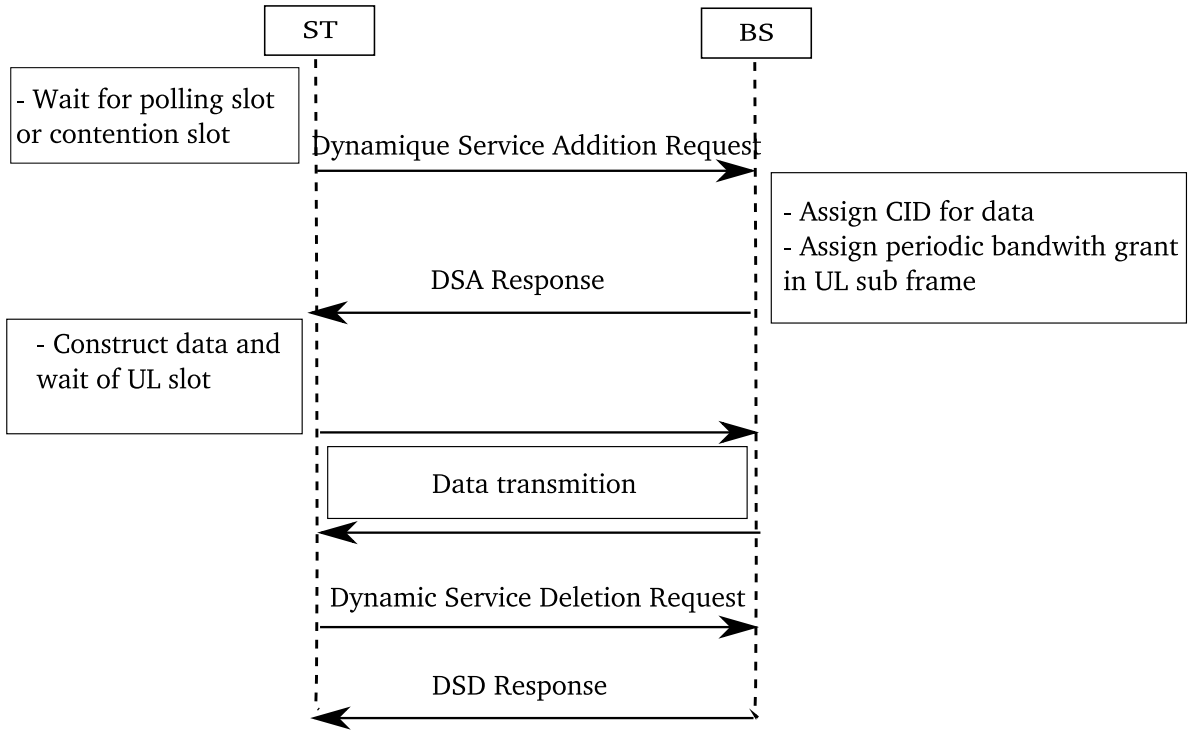


Figure: Sequence diagram of UGS data connection

WiFiRe - Data connection - rtps

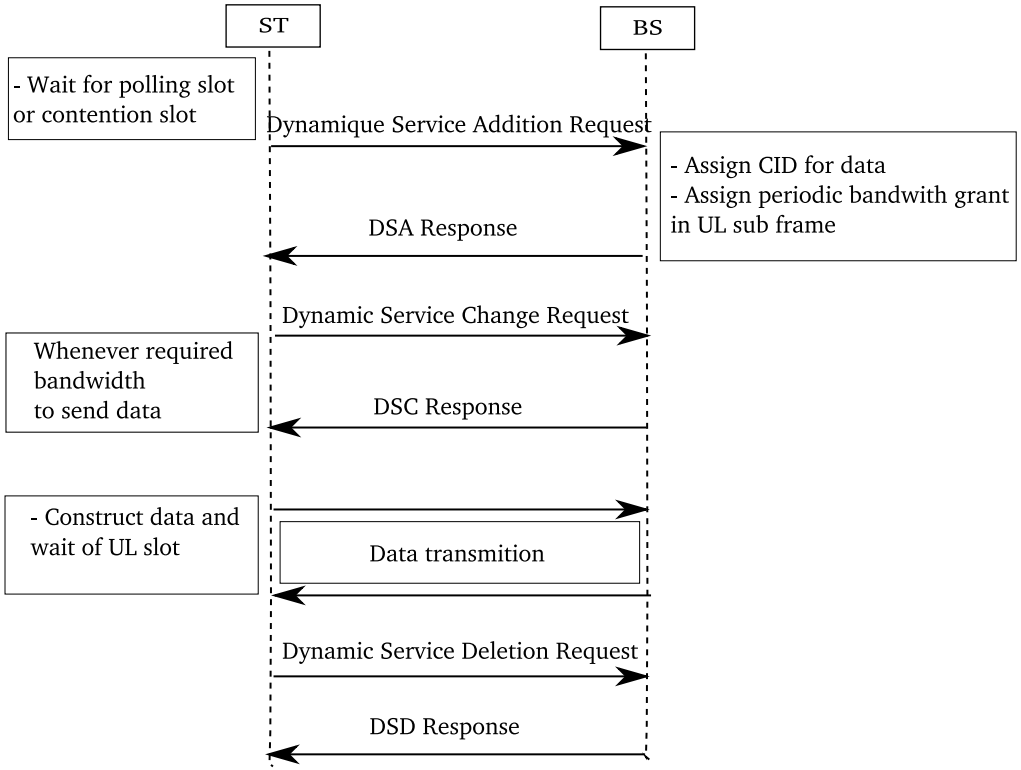


Figure: Sequence diagram of rtPS and nrtPS data connection