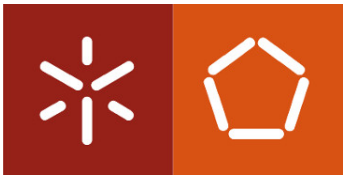

Universidade do Minho

"Implementação de um jogo distribuído numa Rede Veicular (Rede Oportunista/Tolerante a Atrasos) com Dados Nomeados"

Grupo 7

Mestrado em Engenharia Informática e Engenharia de Redes e Serviços Telemáticos

Arquiteturas Emergentes de Rede

pg30520	a84527	pg47245
Diogo Oliveira	Guilherme Soares	Hugo Silva

Braga, 17 de Maio de 2022

1 Conceção e desenvolvimento da aplicação numa rede IPv6 infraestruturada

Numa primeira fase foi decidido o jogo a desenvolver para ser distribuído na rede veícular assim como a sua topologia e configuração.

Após deliberação entre os elementos do grupo fico decidido que o jogo ***Bomberman*** seria o escolhido para o trabalho prático. Este foi desenvolvido em *Python* e, através da capacidade servidor/cliente, permite ser jogado por vários jogadores em simultâneo numa mesma sessão.

1.1 Configuração da Rede

Como topologia de rede utilizamos a seguinte, como se encontra na figura, um servidor, três utilizadores e vários routers conectados por links físicos ou wireless.

De forma a que a comunicação entre a nossa Rede Interna e Routers Móveis "mdr"seja realizada, para que os nodos veiculares/utilizadores consigam aceder ao jogo através da ligação ao servidor e respetiva troca de informações com os routers adjacentes e utilizadores finais na nossa rede, foi adicionado um Router central Wireless "RWireless"a servir de Gateway de comunicação entre todos os dispositivos, tendo-se acrescentado no Zebra a seguinte configuração: ipv6 ospf6 network manet-designated-router. Esta configuração em especifico, possibilitou que existisse endereçamento multicast no routing da nossa rede, ou seja, tornou possível a troca de informações/pacotes entre todas as interfaces ao longo da rede. Nas várias interfaces foram definidas alterações relativamente aos seus Endereçamentos, dentro dos quais nos nossos routers estáticos são representados pelo prefixo de sub-rede /64 enquanto os Routers Wireless, bem como os nós móveis "mdr"são representados pelo prefixo de sub-rede /128, capacitando também desta forma a comunicação entre o nosso Gateway, se não a rede não seria alcançável. Nas várias interfaces foram removidos os Endereçamentos IPv4 e acrescentados IPv6 com a seguinte nomenclatura - "2001:690:2280:xx::x", sendo os "x"são representativos da respetiva Interface de entrada e saída, que se devidamente configurados permitem realizar a troca de informação e routing até outras interfaces e/ou utilizadores finais - como podemos visualizar, nos prints abaixo, quer de pings quer de traceroutes entre routers, routers moveis "mdr"e Utilizadores.

Desta forma, o Routing entre a nossa rede e os nós moveis "mdr"só é possivel de ser realizado através do nosso único Access Point ao qual lhe foi atribuído uma ligação Wireless, capaz de comunicar e fazer redistribuição das nossas rotas até aos utilizadores finais e, com o jogo inicializado, participar e interagir com o jogo que se encontra "à escuta"de um outro dispositivo para dar inicio à participação. Se os nós móveis não se encontrarem ao alcance da rede, ou se estes não conseguirem comunicar entre outros nós móveis, estes não irão conseguir atingir a rede e desta forma não conseguirão comunicar com o jogo. Neste momento, e se devidamente interligados com os "mdrs", os nossos dispositivos de rede são capazes de comunicar entre todos, bem como com o servidor, que se encontra à "escuta"de utilizadores para começarem a jogar.

De seguida apresentamos as configurações do campo ZEBRA Quagga dos diversos tipos de routers existentes.

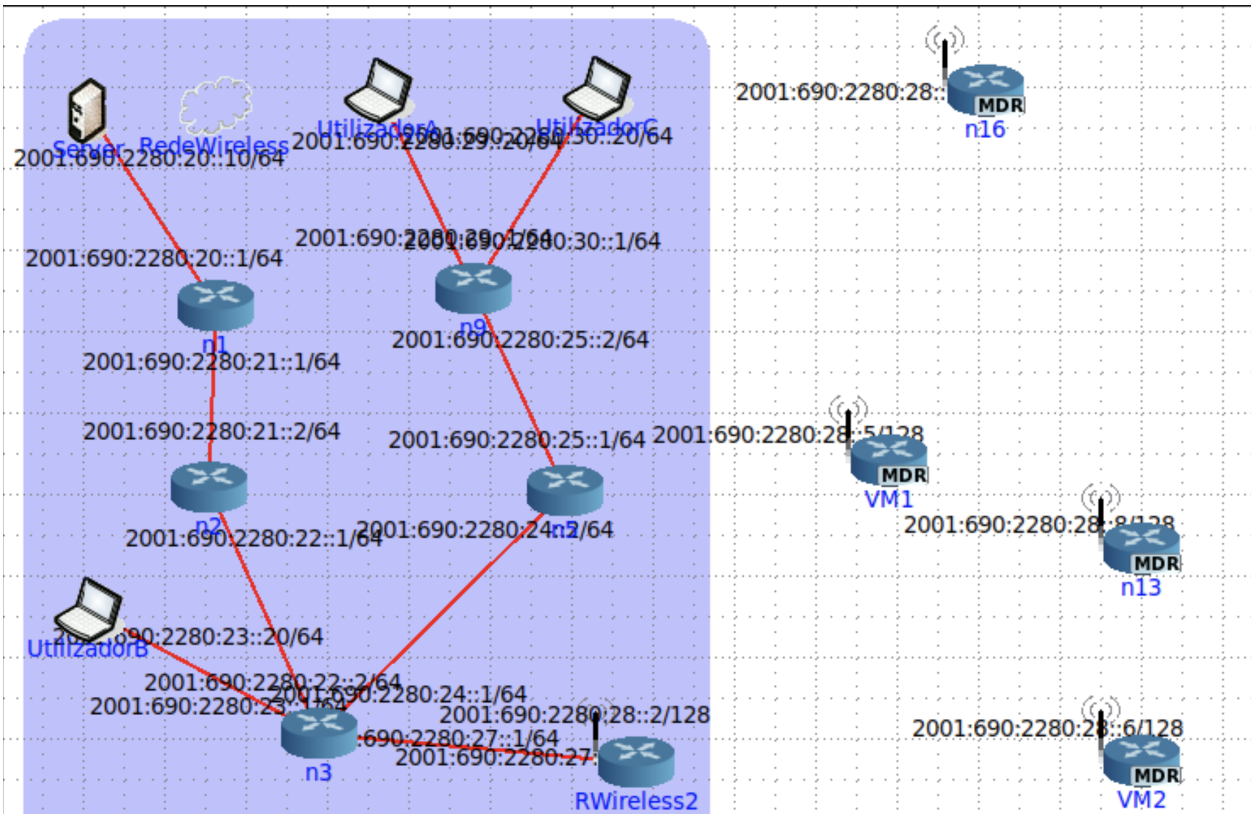

Figura 1: Topologia da rede

```
interface eth0

  ipv6 address 2001:690:2280:25::2/64
  ipv6 ospf6 network broadcast
  ipv6 ospf6 network point-to-point

!
interface eth3

  ipv6 address 2001:690:2280:25::3/64
  ipv6 ospf6 network broadcast
  ipv6 ospf6 network point-to-point
!
router ospf6
  redistribute connected
  router-id 0.0.0.9
  interface eth0 area 0.0.0.0
  interface eth3 area 0.0.0.0
!
```

Figura 2: Configuração Router N9

```
interface eth0

  ipv6 address 2001:690:2280:27::2/64
  ipv6 ospf6 network broadcast
  ipv6 ospf6 network point-to-point
  ipv6 ospf6 adjacencyconnectivity uniconnected
  ipv6 ospf6 lsafullness mincostlsa
!
interface eth1

  ipv6 address 2001:690:2280:28::2/128
  ipv6 ospf6 network broadcast
  ipv6 ospf6 network point-to-point
  ipv6 ospf6 network manet-designated-router
  ipv6 ospf6 adjacencyconnectivity uniconnected
  ipv6 ospf6 lsafullness mincostlsa
!

router ospf6
  redistribute connected
  router-id 0.0.0.4
  interface eth0 area 0.0.0.0
  interface eth1 area 0.0.0.0
!
```

Figura 3: Configuração Router Gateway RWireless2

```
interface eth0

  ipv6 address 2001:690:2280:28::7/128
  ipv6 ospf6 network broadcast
  ipv6 ospf6 adjacencyconnectivity uniconnected
  ipv6 ospf6 lsafullness mincostlsa
!
router ospf6
  redistribute connected
  router-id 0.0.0.20
  interface eth0 area 0.0.0.0
!
```

Figura 4: Configuração Router MDR

1.2 Teste de conectividade IPv6

Através de ping e traceroute conseguimos garantir conectividade IPv6 de ponta a ponta na rede definida, foram realizadas alterações aos routers para que tal seja possível.

Como exemplo apresentamos um ping e um traceroute realizados no UtilizadorB (2001:690:2280:23::20) ao UtilizadorA (2001:690:2280:29::20)

```
<e.43895/UtilizadorB.conf# ping 2001:690:2280:29::20
PING 2001:690:2280:29::20(2001:690:2280:29::20) 56 data bytes
64 bytes from 2001:690:2280:29::20: icmp_seq=1 ttl=61 time=0.502 ms
64 bytes from 2001:690:2280:29::20: icmp_seq=2 ttl=61 time=0.482 ms
64 bytes from 2001:690:2280:29::20: icmp_seq=3 ttl=61 time=0.207 ms
^C
--- 2001:690:2280:29::20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.207/0.397/0.502/0.134 ms
root@UtilizadorB:/tmp/pycore.43895/UtilizadorB.conf#
```

Figura 5: Ping UtilizadorB-UtilizadorA

```
root@UtilizadorB:/tmp/pycore.43895/UtilizadorB.conf# traceroute 2001:690:2280:29::20
traceroute to 2001:690:2280:29::20 (2001:690:2280:29::20), 30 hops max, 80 byte packets
 1  2001:690:2280:23::1 (2001:690:2280:23::1)  0.086 ms  0.054 ms  0.052 ms
 2  2001:690:2280:24::2 (2001:690:2280:24::2)  0.109 ms  0.095 ms  0.095 ms
 3  2001:690:2280:25::2 (2001:690:2280:25::2)  0.265 ms  0.156 ms  0.136 ms
 4  2001:690:2280:29::20 (2001:690:2280:29::20)  0.164 ms  0.153 ms  0.166 ms
root@UtilizadorB:/tmp/pycore.43895/UtilizadorB.conf#
```

Figura 6: Traceroute UtilizadorB-UtilizadorA

1.3 Execução do servidor de jogo

```
core@Server:~/Grupo2122/Bomberman$ python3 server.py
pygame 2.1.2 (SDL 2.0.16, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
UDP server up and listening
```

Figura 7: Servidor em execução

1.4 Execução da aplicação do jogo (jogador 1)

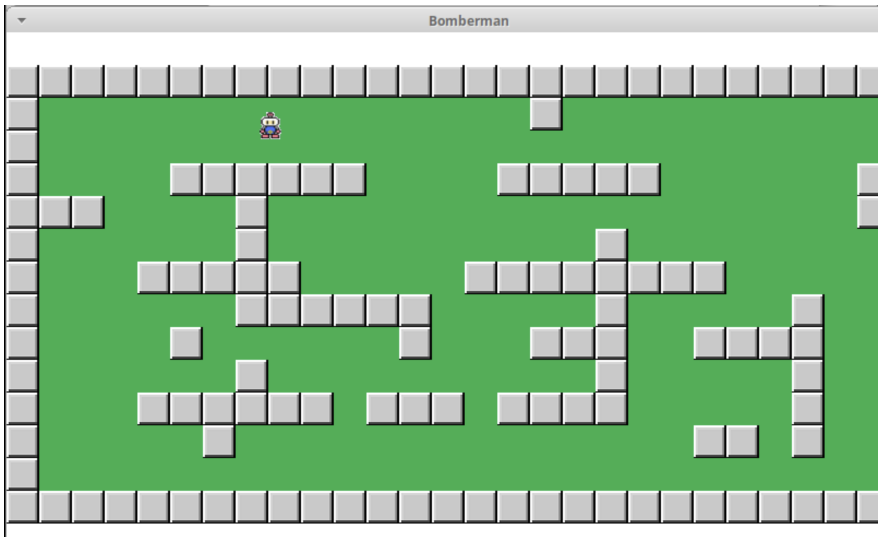

Figura 8: Ecrã de jogo jogador 1

1.5 Execução da aplicação do jogo (jogador 2)

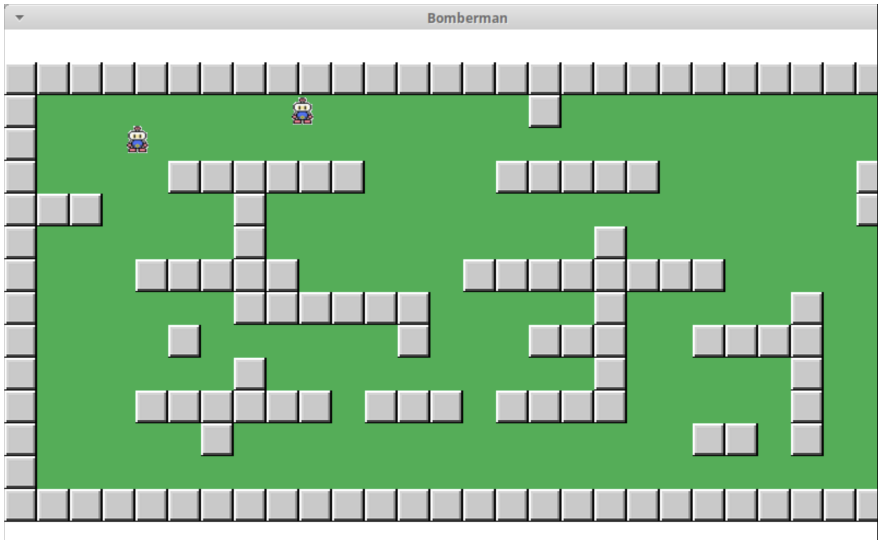

Figura 9: Ecrã de jogo jogador 2

2 Estabelecimento de uma rede veicular ad-hoc, em configuração de rede tolerante a atrasos

2.1 Configuração da rede para incluir os nós móveis (veículos) com suporte ad-hoc nativo em IPv6

Para permitir à rede ter mobilidade adicionou-se à Rede Wireless um ficheiro de cenário que permite simular o movimento dos routers MDR.

O ficheiro "CenarioMobilidadeDTN.scen"contém as posições iniciais (x,y,z) dos routers, com a respetiva movimentação "setdest"e posições finais dos nós/veiculos móveis, assim como os tempos para executar os movimentos.

Figura 10: Ficheiro que garante mobilidade

```
$node_(16) set X_ 580.0
$node_(16) set Y_ 50.0
$node_(16) set Z_ 0.00
$node_(14) set X_ 516.0
$node_(14) set Y_ 250.0
$node_(14) set Z_ 0.00
```

Figura 11: Definição da posição inicial de dois routers

2.2 Configuração da rede para incluir o movimento dos veículos

```
$ns_ at 3.00 "$node_(16) setdest 300.0 278.0 35.0"
$ns_ at 10.00 "$node_(16) setdest 350.0 358.0 15.0"
$ns_ at 25.00 "$node_(16) setdest 180.0 448.0 35.0"
$ns_ at 20.00 "$node_(16) setdest 280.0 248.0 35.0"
$ns_ at 2.00 "$node_(14) setdest 150.0 288.0 25.0"
$ns_ at 8.00 "$node_(14) setdest 250.0 150.0 20.0"
$ns_ at 15.00 "$node_(14) setdest 416.0 100.0 20.0"
$ns_ at 7.00 "$node_(14) setdest 516.0 60.0 30.0"
```

Figura 12: Atribuição de movimentos a dois routers

Após colocar no ficheiro ".scen"os movimentos de cada router, como demonstrado na imagem acima, é possivel ativar na topologia o movimento dos veículos. Começamos por adicionar o ficheiro à rede Wireless para que o utilize para definir os movimentos dos veículos e aquando da ativação da topologia é possível correr o ficheiro e assim observar o movimento nos veículos.

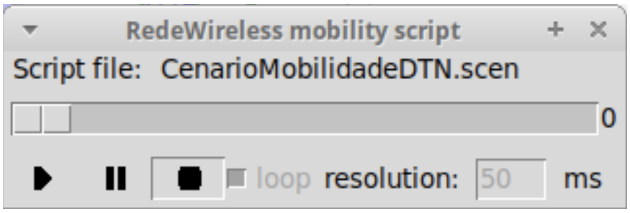

Figura 13: Janela de ativação do movimento

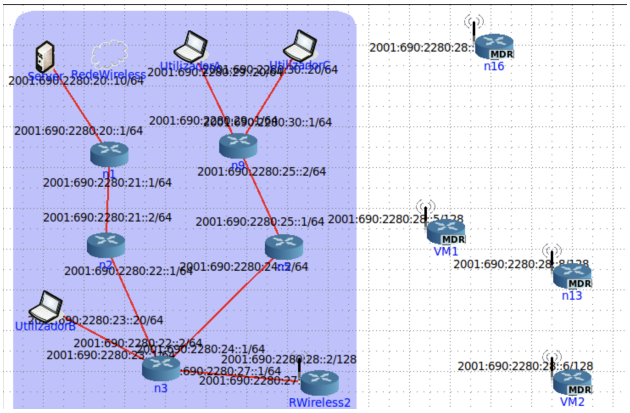

Figura 14: Posição inicial dos veículos

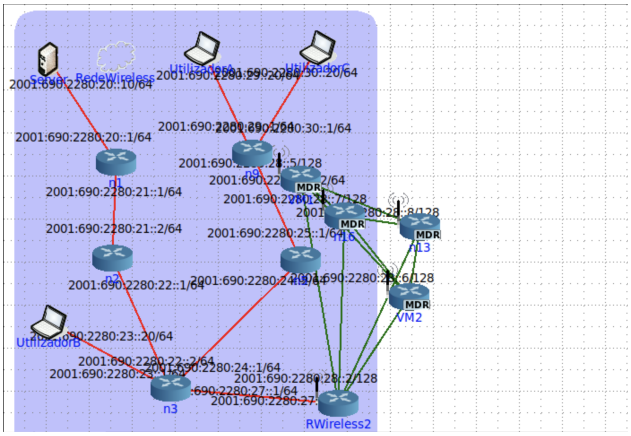

Figura 15: Posição intermédia dos veículos

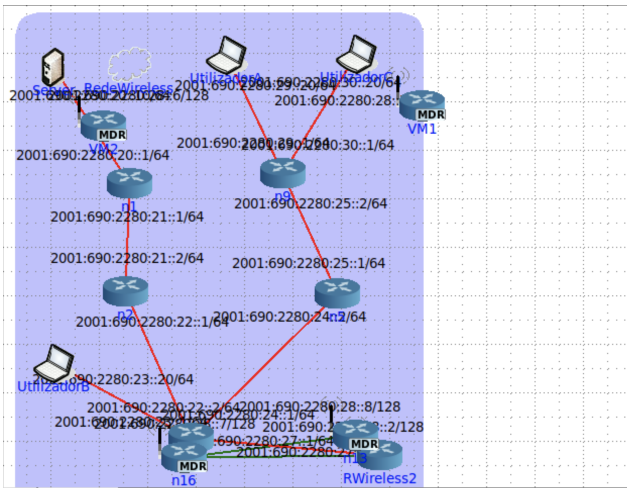

Figura 16: Posição final dos veículos

2.3 Testar a conectividade em IPv6 nos nós móveis;

Através de ping e traceroute conseguimos garantir conectividade IPv6 de ponta a ponta na rede definida com os nós móveis, foram realizados pings e traceroutes para comprovar essa conectividade.

Os pings e traceroutes efetuados foram desde o UtilizadorB (2001:690:2280:23::20/64) ao nodo VM1 (2001:690:2280:28::5/128

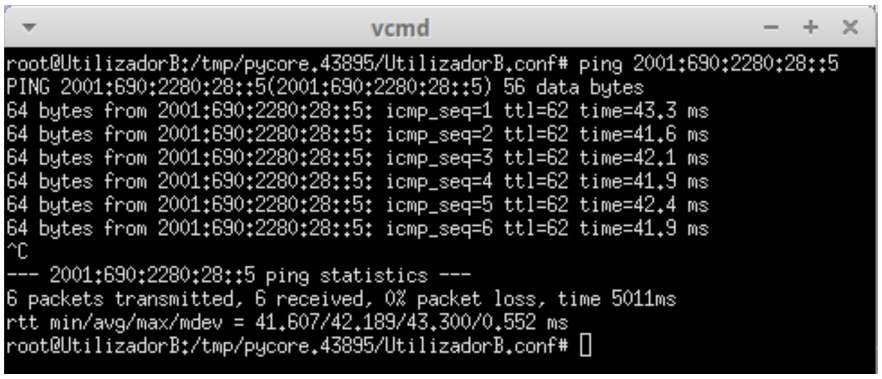

Figura 17: Ping UtilizadorB-VM1

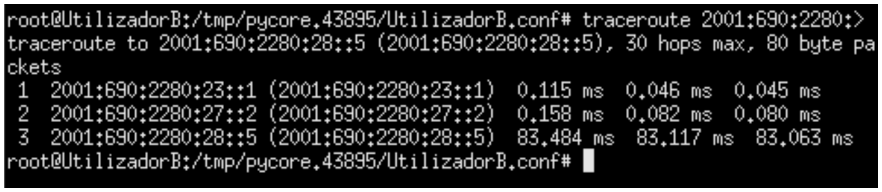

Figura 18: Traceroute UtilizadorB-VM1

2.4 Implementação da estratégia de encaminhamento DTN para os nós móveis

Foi criada uma classe *WirelessNode*, classe essa que tem como atributos o IP; porta; uma *flag* que indica se é *gateway* ou não e uma lista de vizinhos. Quanto a métodos, a classe tem um método que vai ficar a enviar e a receber mensagens *multicast* ,através de um socket específico, de forma a conhecer os vizinhos que tem e atualizar a lista de vizinhos. Cada um desses vizinhos terá associado um *score* para determinar o melhor vizinho. Existe outro método que, caso o nodo seja um *gateway*, irá ser chamado, abrindo uma socket para comunicar com o *server*, sendo que se for *gateway* vai existir uma comunicação constante entre esse e o *server* visto que o *gateway* mal receba informação de um nodo wireless vai enviar para o *server* e mal receba informação do *server* vai mandar para os nodos wireless que tem que mandar.

2.5 Implementar, instalar e executar esse novo módulo em todos os veículos

Dado que os nós móveis vão ter um IP específico é apenas necessário correr o *script* nesse nó, *script* esse que vai estar a mandar mensagens *multicast* de presença e a receber consoante encontra nós no range do wifi atualizando assim a sua tabela de vizinhos. Enquanto a tabela vai sendo atualizada, um algoritmo calcula o melhor nó a escolher para enviar a mensagem do jogo. Essa mensagem eventualmente chegará ao *gateway* de onde seguirá para o servidor. O servidor por sua vez atualizará o estado do jogo consoante a alteração que recebeu e envia a atualização para o *gateway* que irá distribuir pelos clientes até estes receberem com sucesso a mensagem, isto por ser DTN, *delay tolerant*, pois podem não estar conectados no momento do envio mas assim que estejam recebem.

2.6 Execução do servidor de jogo

```
core@Server:~/Grupo2122/Bomberman$ python3 server.py
pygame 2.1.2 (SDL 2.0.16, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
UDP server up and listening
```

Figura 19: Servidor em execução

2.7 Execução da aplicação do jogo no veículo 1 e no veículo 2

Aquando da execução do jogo nos nós móveis dentro da topologia deparamos-nos com erros que não conseguimos resolver como apresentado nas imagens seguintes.

```
root@n16:/tmp/pycore.34111/n16.conf# su - core
core@n16:~$ cd Grupo2122/
core@n16:~/Grupo2122$ cd  Bomberman/
core@n16:~/Grupo2122/Bomberman$ export DISPLAY=:0.0
core@n16:~/Grupo2122/Bomberman$ python3 game.py 2001:690:2280:20::10
pygame 2.1.2 (SDL 2.0.16, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
ALSA lib confmisc.c:767:(parse_card) cannot find card '0'
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_card_driver return
ed error: No such file or directory
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_concat returned er
ror: No such file or directory
ALSA lib confmisc.c:1246:(snd_func_refer) error evaluating name
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_refer returned err
or: No such file or directory
ALSA lib conf.c:5220:(snd_config_expand) Evaluate error: No such file or directo
ry
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM default
```

Figura 20: Erro apresentado Cliente

Figura 21: Bomberman Blackscreen n13

Figura 22: Bomberman Blackscreen n16

3 Conclusão

Acreditamos que o trabalho inicialmente planeado por nós, apesar de termos implementado a nossa rede como o esperado e todos os dispositivos de rede comunicarem entre si bem como a inicializarão do jogo, não correu como o esperado visto que pretendíamos uma arquitetura mais complexa dentro da nossa rede interna, que devido a entraves do core nos impossibilitaram de o realizar. Dado que não nos foi possível terminar a tempo, pretendemos com trabalho futuro finalizar os pontos menos bem conseguidos que sabemos como solucionar, como é o caso da implementação da estratégia de encaminhamento DTN, de forma a que fosse de encontro aos objetivos totais deste trabalho.