Trabalho Prático Nº2 – Desenho e implementação de um serviço de mensagens em rede AdHoc

Duração: 6 aulas (não consecutivas, ver calendário)

Motivação

Numa rede estruturada, praticamente todos os nós implementam a pilha protocolar completa, desde os routers aos sistemas finais. Os routers, cujas funções principais são da camada de rede (layer3), quase poderiam dispensar o transporte e as aplicações, não fora dar-se o caso de precisarem ser configurados remotamente, precisamente usando protocolos de aplicação como o telnet ou ssh! Logo precisam da pilha toda. Os hosts, que só precisam de enviar e receber os seus próprios pacotes, prescindem dos protocolos de routing da camada 3, usando apenas rotas estáticas e um default router.

Mas numa rede ad-hoc, de formação espontânea, não há nós com papel especial e todos precisam de ajudar a fazer tudo. Mesmo um simples telemóvel, tem de colaborar no envio e recepção de pacotes de outros telemóveis, se quiser colaborar na formação de uma rede funcional.

Neste trabalho pretende-se implementar um pequeno protótipo de uma rede de formação espontânea (AdHoc), em que os nós são capazes de descobrir rotas para outros nós com ajuda dos seus vizinhos. O serviço a fornecer é simplesmente um serviço de mensagens curtas, tipo twitter. Pode-se "twittar" um nó específico. Nesta rede os endereços IP são de pouca utilidade. Só os nomes interessam, pois é a eles que se dirigem as mensagens curtas.

Objectivos

O principal objectivo deste trabalho é desenhar e implementar um serviço de mensagens curtas (tamanho máximo bem determinado) sobre IPv6. Serão usados dois protocolos diferentes: um para descobrir rotas e outro para enviar e receber as mensagens propriamente ditas.

Descrição

Em cada nó coloca-se em execução uma instância de uma aplicação AdHoc, designada por *adhoc_app*. Ao iniciar, a aplicação *adhoc_app* não conhece nada nem ninguém à sua volta. Fica de imediato à escuta na porta 9999 em TCP, para receber e enviar mensagens curtas dos utilizadores. E na mesma porta 9999 em UDP para dialogar com os nós vizinhos de acordo com o protocolo a implementar. A primeira operação é descobrir vizinhos, sem a qual nada de útil se pode fazer. Constrói-se uma mensagem HELLO, e envia-se por multicast para o endereço IPv6 de grupo FF02::1 ("all hosts"), com TTL=1 (só vizinhos directos). Se todos os nós enviarem e escutarem estas mensagens na porta 9999, ficam a conhecer, de imediato, os vizinhos directos (V=1). Se nas mensagens incluírem como informação adicional a lista dos seus vizinhos conhecidos, todos ficam a conhecer integralmente a vizinhança de raio 2 (V=2). Esta tabela de rotas é uma espécie de "olho do peixe" (analogia com a imagem da figura 1) do nosso protocolo e permitirá resolver a maior parte das comunicações sem mais demoras.

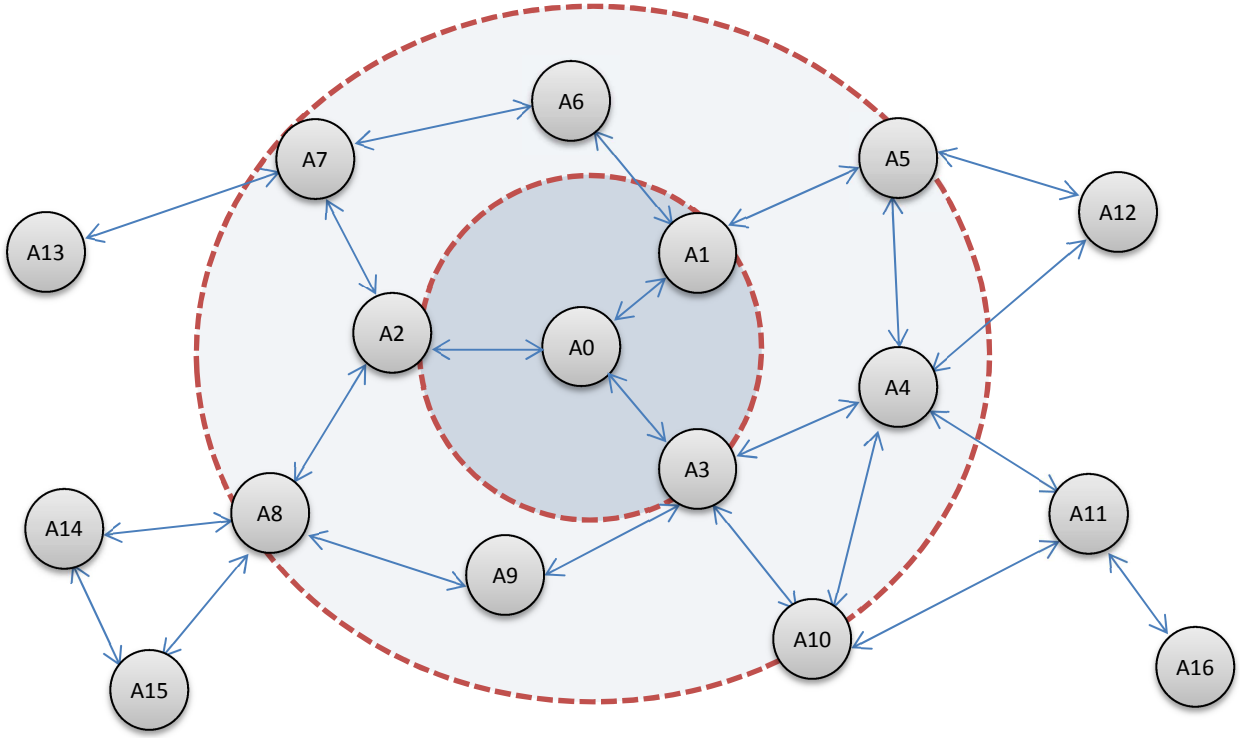


Figura 1

A título de exemplo apresenta-se a tabela de encaminhamento do nó *App0* da figura 1:

Nome	Vizinho	Endereço do vizinho
A1	A1	IPv6(A1)
A2	A2	IPv6(A2)
A3	A3	IPv6(A3)
A4	A3	IPv6(A3)
A5	A1	IPv6(A1)
A6	A1	IPv6(A1)
A7	A2	IPv6(A2)
A8	A2	IPv6(A2)
A9	A3	IPv6(A3)
A10	A3	IPv6(A3)

Mensagens recebidas pelo nó que sejam dirigidas a algum vizinho da vizinhança V=2, podem ser entregues sem dificuldade, direta ou indiretamente. Quanto aos nós fora dessa vizinhança, caso existam e estejam alcançáveis, serão descobertos com pedidos de rota especiais ROUTE REQUEST. Os pedidos devem ser difundidos de todos para todos, até uma vizinhança máxima incluída no pedido de Z=N; Se o pedido atingir um nó que possui uma rota válida, ou o próprio nó de destino, este responde com um ROUTE REPLY. A mensagem de ROUTE REPLY terá de conseguir fazer o percurso inverso à mensagem de ROUTE REQUEST, e as tabelas de encaminhamento nesse percurso. Se ao fim de algum tempo nenhuma resposta for recebida, o nó destino deve ser declarado temporariamente inatingível.

Naturalmente que pedidos muito frequentes de rota para fora da vizinhança V=2 podem originar muito tráfego na rede, eventualmente desnecessário. Mas pode-se tirar partido da cache de rotas e minimizar pedidos frequentes. Um exemplo seria guardar a não existência de rota (cache negativa). Mas recomenda-se que idealizem e proponham melhorias a este processo de modo a torná-lo mais simples e eficiente.

Relativamente aos protocolos a desenvolver incluem-se abaixo algumas sugestões concretas. Os grupos têm liberdade para decidir e justificar toda e qualquer escolha que façam no design do seu serviço.

Protocolo HELLO (8 valores, 2 aulas)

Parâmetros de configuração recomendados:

- *hello interval* - período de tempo entre dois HELLO consecutivos do mesmo nó;
- *dead interval* – período de tempo de silêncio de um nó a partir do qual ele é declarado inalcançável

As mensagens de HELLO devem ter um formato bem definido e simples. Nos dados podem incluir a lista de vizinhos de vizinhança V=1 que conhecem (com o cuidado de evitar ou lidar bem com a fragmentação); O endereço IPv6 de destino é FF02::1, porta 9999 (UDP), TTL=1.

Protocolo ROUTE_REQUEST (6 valores, 2 aulas)

Parâmetros de configuração sugeridos:

- *timeout* – tempo máximo de espera por uma resposta a um pedido de rota...
- *radius* – raio máximo para procurar por uma rota (em numero de saltos)

As mensagens ROUTE_REQUEST despoletam um processo de descoberta de rota para um determinado destino. O nó de destino, ou um nó com uma rota válida na tabela de vizinhança V=2 pode responder ao pedido com uma mensagem ROUTE_REPLY. Estas mensagens também devem ser enviadas e recebidas na porta 9999 (UDP).

Protocolo MESSAGE/TW (6 valores, 2 aulas)

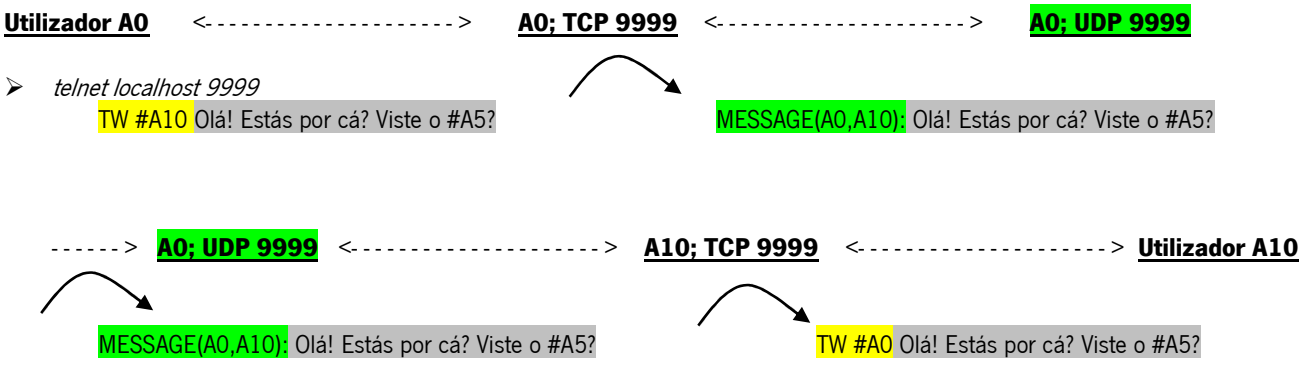
Parâmetros de configuração requeridos:

- *message size* – tamanho máximo de mensagem (menor que 500)

Este protocolo é o equivalente da camada de aplicação, sendo pois formato texto, legível para o utilizador. Não é necessário nenhum cliente especial, pois um utilizador experiente pode usar o *telnet* para a porta TCP 9999 e escrever os comandos ou ler as mensagens. Sugestões:

TW #A10 Olá #A10! Estás por cá? Viste o #A5?

Estas mensagens são entregues assim diretamente pelo utilizador na porta TCP 9999. O servidor TW recebe-as do utilizador no formato TW definido, acrescenta-lhes um cabeçalho MESSAGE com o nó de origem (o seu próprio nome!) e nó destino (o nome escrito logo após o TW, #A10 no exemplo) e entrega-as na porta UDP 9999 em *localhost*. As mensagens são depois reencaminhadas pelos agentes UDP, nó a nó até ao destino. O agente UDP do nó destino retira o cabeçalho MESSAGE e entrega a mensagem na porta TCP 9999 local.



O protocolo MESSAGE, que encapsula as mensagens TW, também pode usar formato texto muito simples. Exemplo: MESSAGE (#from,#to) "<message>".

Planeamento

O desenho e implementação deste serviço devem ser feitos ao longo das aulas práticas de CC, quer nas explicitamente dedicadas ao trabalho, como em todas as outras onde existir tempo livre. As fases de desenvolvimento são as seguintes:

FASE 1:

AULA 3 – Semana de 10 a 14 de Março – Apresentação e início da fase1;
AULA 4 – Semana de 17 a 21 de Março – Continuação da fase 1;

A fase 1 é dedicada à implementação do protocolo HELLO. O cenário de teste e de demonstração é o seguinte:
(...)
8 valores; Avaliação contínua;

FASE 2:

AULA – Semana de – Início da fase2;
AULA – Semana de – Início da fase2;

A fase 2 é dedicada ao desenho e implementação do protocolo ROUTE_REQUEST. O cenário de teste e de demonstração é o seguinte: (...)

6 valores; Avaliação contínua;

FASE 3:

AULA – Semana de – Início da fase2;
AULA – Semana de – Início da fase2;

A fase 3 é dedicada ao desenho e implementação do protocolo MESSAGE/TW.
6 valores; Avaliação contínua; Com demo final e mini relatório;

Relatório

O relatório deve ser escrito em formato de artigo com um máximo de 8 páginas (recomenda-se o uso do formato LNCS - *Lecture Notes in Computer Science*, instruções para autores em http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0). Deve descrever o essencial do desenho e implementação com a seguinte estrutura recomendada:

- Introdução
- Especificação do protocolo
  * primitivas de comunicação
  * formato das mensagens protocolares (PDU)
  * interacções

- implementação
  * detalhes, parâmetros, bibliotecas de funções, etc.
- testes e resultados
- conclusões e trabalho futuro

NOTA: As fases 2 e 3 só podem ser efectuadas depois da fase 1...