# END-TO-END ARGUMENTS IN SYSTEM DESIGN (SRC84)

**Grupo 1**
Francisco Caeiro, **47823**
Bruno Andrade, **47829**
António Estriga, **47839**

**Qual é o problema que os autores tentam resolver?**

À medida que os sistemas foram evoluindo nas décadas de 70 e 80 foi-se debatendo qual seria a melhor maneira que garantisse uma comunicação cada vez mais fiável entre máquinas. Estas garantias eram feitas através do uso de funções nas camadas de comunicação, quer ao **nível da aplicação** (alto nível) quer ao **nível de rede** (baixo nível).

Este artigo vem tentar arranjar regras e práticas usando argumentos contra o uso de subsistemas de baixo nível de maneira a facilitar e melhorar a comunicação ponta-a-ponta (*end-to-end*), evitando funções redundantes ou possíveis perdas de desempenho.

**Este problema é relevante?**

Na década de 80, foi começado a ser desenvolvido o *OSI Model*, um modelo de interconexão entre sistemas, dividido em 7 níveis, onde cada um deles tinha um papel especializado. Este artigo veio ajudar o desenvolvimento deste modelo, argumentando onde é que as funcionalidades deveriam ser implementadas, nomeadamente funções de cifra ou garantias de envio e entrega de pacotes.

**Qual é a sua solução?**

A solução proposta varia dependendo de cada protocolo ou função que se quer utilizar, mas segue um princípio de especialização tendo em conta o desempenho, isto é, as funções necessárias deverão ser utilizadas pelas aplicações que as necessitem, de maneira a não prejudicar outras aplicações que não façam uso das mesmas funções e causar um *throughtput* menor.

**Que novas técnicas foram usadas?**

Para fundamentar o argumento *end-to-end*, foram dados vários exemplos de problemas reais e de como a sua solução poderia ser diferente e as razões do mesmo.

No primeiro caso, perdas ou danos de dados, foi sugerida um reenvio total da informação. A deteção das perdas ou danos é feita utilizando *checksums*. Se não existir diferenças entre o *checksum* recebido e o calculado, dá-se a transferência como bem-sucedida, caso contrário envia-se toda a informação novamente. No entanto nem todas as aplicações necessitam de todos os dados corretos (ex. VOIP).

No segundo caso fala sobre as garantias de envio e entrega. Eram utilizados protocolos de *acknowledgement* (ACKS) para dar garantias que a informação chegou corretamente ao recetor. Mais uma vez nem todas a aplicações necessitam deste controlo.

No terceiro caso são apresentados três argumentos para a cifra de dados. Para respeitar os argumentos o mais lógico seria usar as funções ao nível da aplicação, visto que nem todos os datagramas que passam pela rede necessitam de ser cifrados. No entanto, a cifra de dados também pode ser feita a um nível mais baixo o que é menos sofisticada.

O quarto caso aborda a supressão de mensagens duplicadas. Como cada aplicação deverá ter em conta a receção ou envio de mensagens duplicadas por diversas razões, faz todo o sentido que o tratamento de tais mensagens seja feito no nível de aplicação e não ao nível de rede, visto que a rede não sabe que mensagens duplicadas são necessárias ou não.

O quinto caso fala sobre o envio e entrega de mensagens pela ordem FIFO dos dois lados, ou seja, se o emissor enviou $M_a$ -> $M_b$ -> $M_c$ então o recetor irá entregar as mensagens pela mesma ordem, independentemente da ordem de como as recebeu.

No sexto caso o problema das mensagens duplicadas levanta um novo problema, escritas e leituras duplicadas. Num sistema de transações quando existe uma operação de escrita a informação é escrita em disco, o que, no caso de existirem pedidos duplicados de escrita, diminuem a performance devido ao custo da escrita em disco. No caso de escritas duplicadas, a 2ª mensagem do pedido é descartada. O mesmo já não acontece no caso das leituras. Nas operações de leitura, o problema de *acknowledgement* é redundante, visto que o envio da leitura é prova suficiente do sucesso da leitura. Estas duas abordagens são suficientes para diminuir a quantidade de mensagens na rede.

**Como é que se destaca de trabalhos anteriores?**

Este artigo tenta chamar a atenção ao elevado custo de certas operações em níveis mais baixos, que nem sempre trazem os benefícios necessários para justificar a sua utilização devido a um grande *overhead* de recursos.

**Quais são os pontos mais fortes deste artigo? E os seus pontos fracos?**

O ponto mais forte deste artigo é a mudança de paradigma da rede. Este artigo ajudou bastante a melhorar o desempenho e desenvolvimento das redes, movendo o máximo de funcionalidade para as pontas das redes, ou seja, os altos níveis dos sistemas. Ao evitar que os pontos intermédios da rede, tais como os routers, tenham de garantir a fidelidade e integridade dos dados, sabemos que se manterão simples e não farão muito mais do que o seu trabalho original. Assim, a inovação pode ocorrer com mais facilidade e com menos custos quando é feito nas pontas, pois os dispositivos a alterar são mais tangíveis e em menor quantidade.

Em contraste, como ponto fraco, não é detalhado o paradigma de *layers* que hoje existe na rede, isto é, centram-se em dar ideias de onde fazer as operações, mas não dão nenhuma *guidelide* explícita por cada módulo. Outro defeito encontrado é a explicação de conceitos simples ser, por vezes, demasiado longa e complexa.

**Como seria uma extensão deste trabalho?**

Tendo em conta que este *paper* foi publicado em 1984, podemos atualmente ver trabalho derivado deste argumento, por exemplo, no protocolo de aplicação TLS/SSL, usado para garantir a segurança de transmissão de dados. Apesar de já criados antes da publicação deste *paper*, podemos também estudar o exemplo do protocolos TCP e UDP, usados na camada de transporte e que garantem, ou não, a integridade dos pacotes enviados.

Tal como mencionado anteriormente, a primitiva de orientar a funcionalidade da rede para as pontas da mesma ajudou a moldar a Internet assim como todas as redes de comunicação informáticas. Sem o estudo deste argumento, o crescimento de ambos como não seria tão rápido e económico como foi, e continua a ser.