Primeiramente, devemos começar falando sobre a tecnologia que foi a base para a criação dos primeiros middlewares: Os sockets. Criado nos anos 70 pela ARPANET como um dos primeiros processos de comunicação entre computadores. É baseado no modelo cliente-servidor, onde dados são transmitidos por meio de buffers ou fluxo de bytes. Foi um verdadeiro marco para o começo do desenvolvimento das tecnologias de comunicação entre máquinas, mas logicamente, tinha desvantagens claras de uso como o baixo nível de abstração (e necessidade de conhecimento de redes) e ter seu código muito atrelado ao código das aplicações. A partir dos sockets, foram desenvolvidas as primeiras ferramentas de programação distribuída: RMI e RPC, que livraram do programador a necessidade de lidar diretamente com endereçamento e (de)codificação de mensagens, além ser capaz de permitir a invocação de métodos em objetos remotos, algo inovador até então. Nessas aplicações, os códigos que envolvem as chamadas a rede foram mascarados por procedimentos chamados Stubs, que realizam todo processo de passagem de parâmetros entre procedimentos. Tal tecnologia, ainda oferece um serviço de nomes, que serve para facilitar a localização de objetos e métodos. Em relação a suas desvantagens, por fazerem parte da primeira leva de middleware, possuíam uma quantidade limitada de serviços de plataforma, além de serem bem presas ao uso de uma única linguagem de programação. Após elas, vieram evoluções como o CORBA, que seguiam um modelo de funcionamento similar, mas apresentavam uma quantidade maior de serviços, além de não ser mais amarrada a uma linguagem de programação específica. A partir da segunda leva de sistemas distribuídos, a arquitetura no modelo cliente-servidor (que era dominante), passou a dar lugar a arquitetura orientada a serviços. Tal modelo de implementação, baseado na disponibilização de conjuntos de operações para serem requisitadas pelos clientes, foi capaz de aumentar mais ainda o nível de abstração desse tipo de sistema distribuído, além de garantir interoperabilidade e reusabilidade, e ser completamente isolada da interface. Outra novidade trazida por tipo de arquitetura, se deu com a composição de serviços, proporcionando uma certa facilidade de trabalho para o programador, na parte de reuso de código. Mas, como nem tudo são flores, novos problemas passaram a surgir com o advento desse tipo de tecnologia. Com a possibilidade de distribuir cada vez mais seus sistemas em objetos remotos, o desafio de manter a coordenação e garantia de funcionamento se tornou cada vez maior. Além disso, a notação WSDL não provia uma descrição semântica das funcionalidades, impossibilitando o entendimento por parte dos agentes de software. Dentro da categoria dos serviços de rede, houveram duas gerações: Primeiramente o SOAP, cuja concepção era baseada principalmente em simplicidade, e era com conexão. Depois dele veio o REST, que se caracterizou principalmente por seu forte desacoplamento, ser menos verboso que seu antecessor, e utilizando o HTTP com operações sem estado, de forma a não sobrecarregar o servidor.