Constructing VPNs in Pathlet Routing

Brighten Godfrey

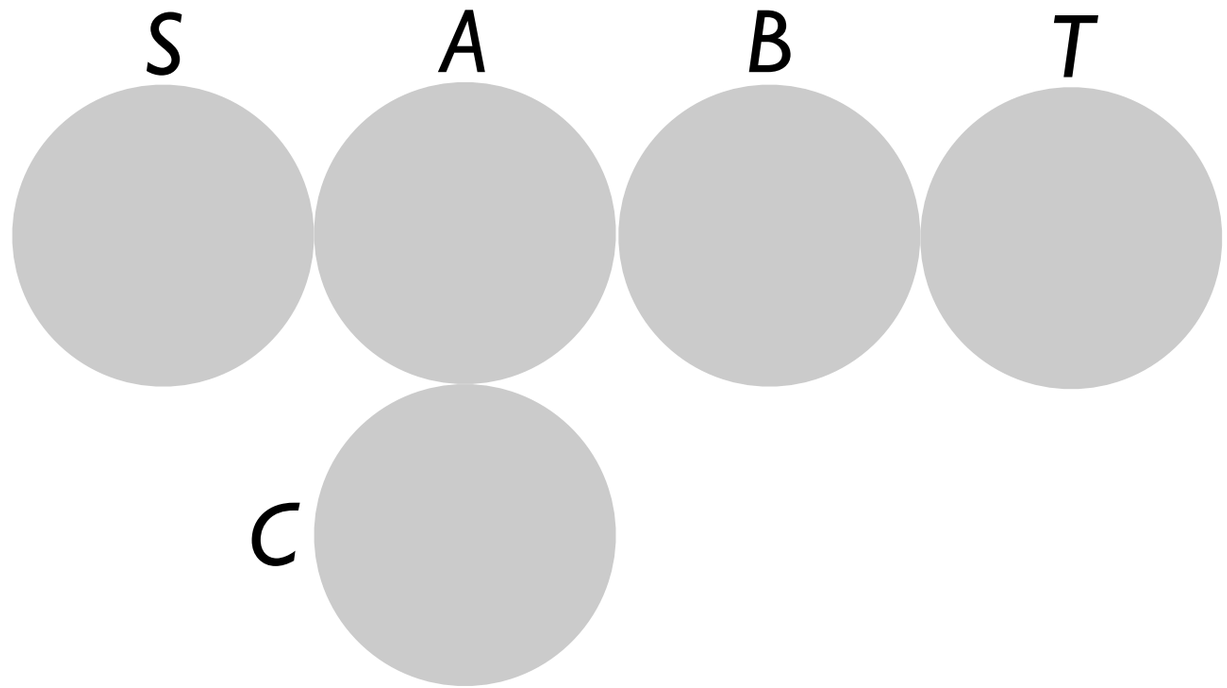
July 1, 2009

Revised Oct 9, 2009

Thanks to Nick Feamster, Nick McKeown, Guru Parulkar, Jennifer Rexford, and Scott Shenker, whose discussions posed the question of how to support VPNs.

VPNs in Pathlet Routing

Suppose we have this AS-level topology and we want to set up a VPN between *S* and *T*, so that so they can reach each other, but other nodes (in particular, *C*) cannot send to or receive from them. Unlike the typical case in the Internet today, this VPN spans multiple providers (*A* and *B*).

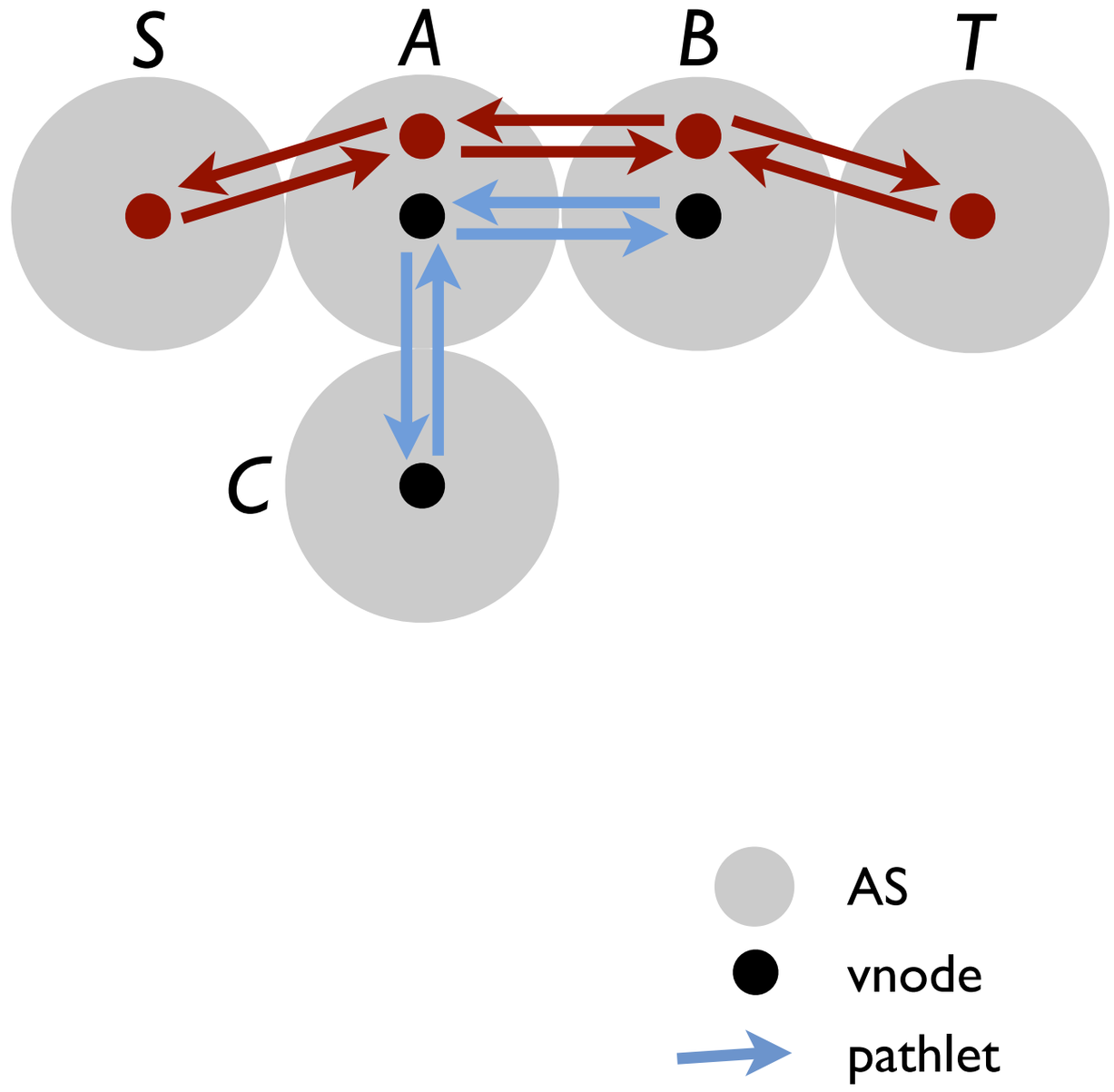


Meanwhile, all the nodes in the interior (*A*,*B*,*C*) should be able to communicate freely.

We'll assume nodes along the VPN path cooperate.

What we basically want

We *basically* want the vnodes and pathlets shown at right. *S* and *T* have their private connection not accessible by *C* even though *C* can route to, and through, *A* and *B*. As you can see, it's easy to construct a virtual private network, since pathlet routing effectively routes on a virtual topology.



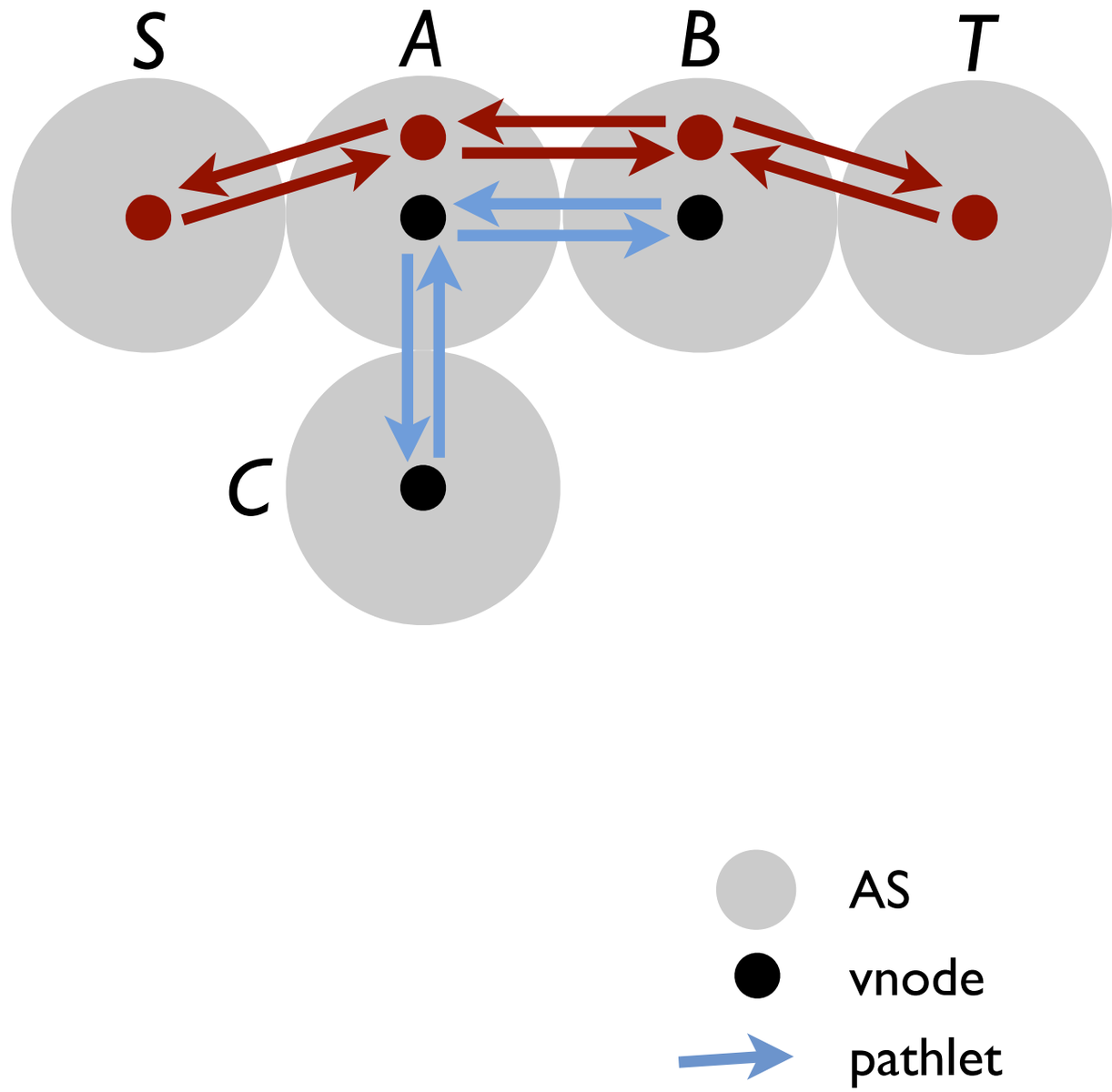
As usual in pathlet routing, this policy is enforced strongly in the data plane: there is no sequence of bits that *C* can put in a packet header that would cause it to arrive at *S* or *T*.

But we're glossing over one detail, dealing with what we call ingress vnodes. For example, *B* should be able to send to both of *A*'s vnodes, but *C* should only be able to send to the black one.

Ingress vnodes

Logically, *A* exposes a set of ingress vnodes to S and *B* (namely, the set is both of its vnodes), but *A* exposes only the black vnode to *C* because *C* is not permitted access to the VPN. When *B* sends *A* a packet, it tags the packet with the intended next-hop vnode. If *C* sends *A* a packet tagged with the red vnode, the packet is dropped.

At a high level, that's all there is to it. You can now stop reading unless you want the nitty-gritty details.

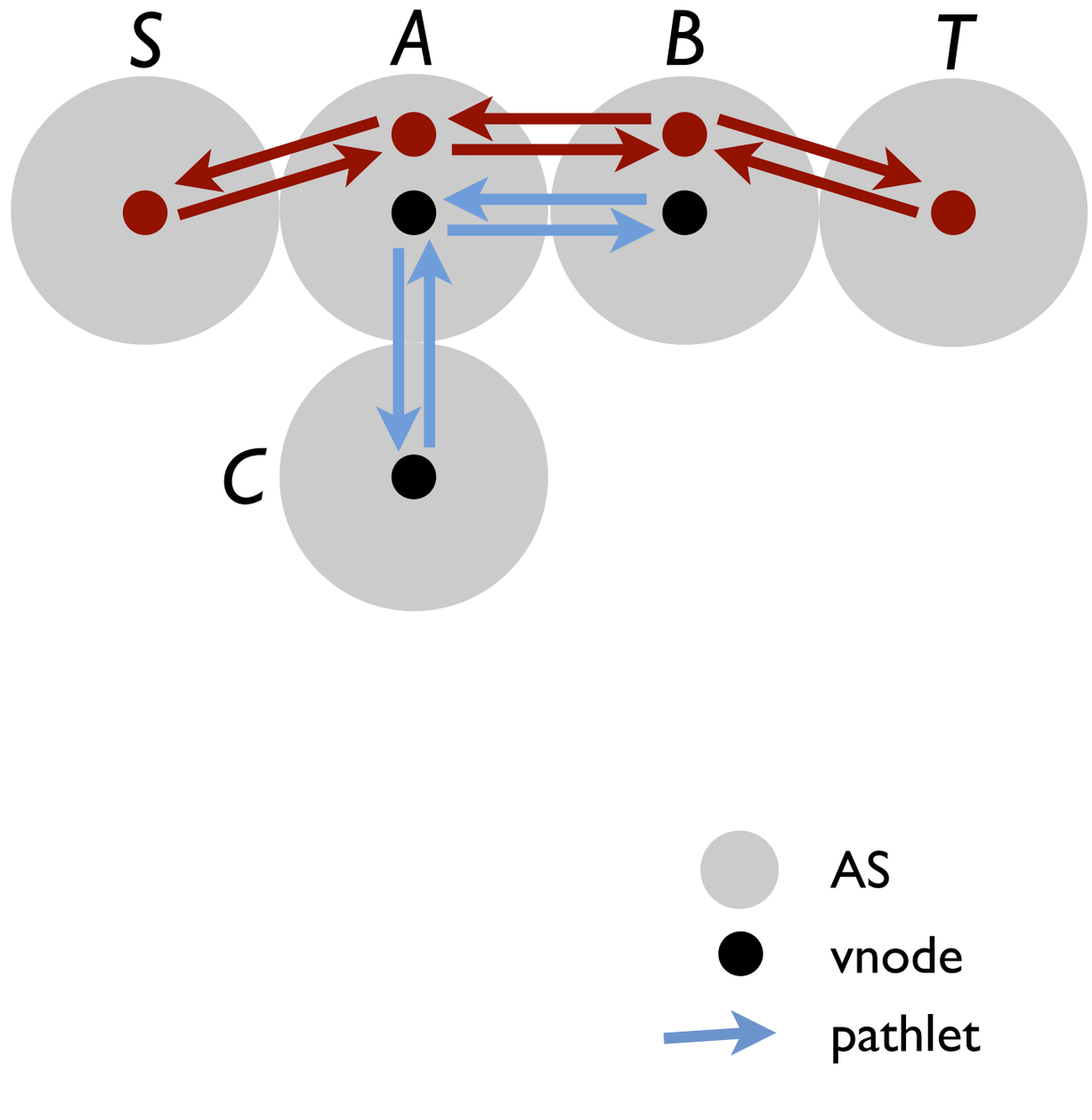


The details

The rest of this document is about the nitty-gritty details.

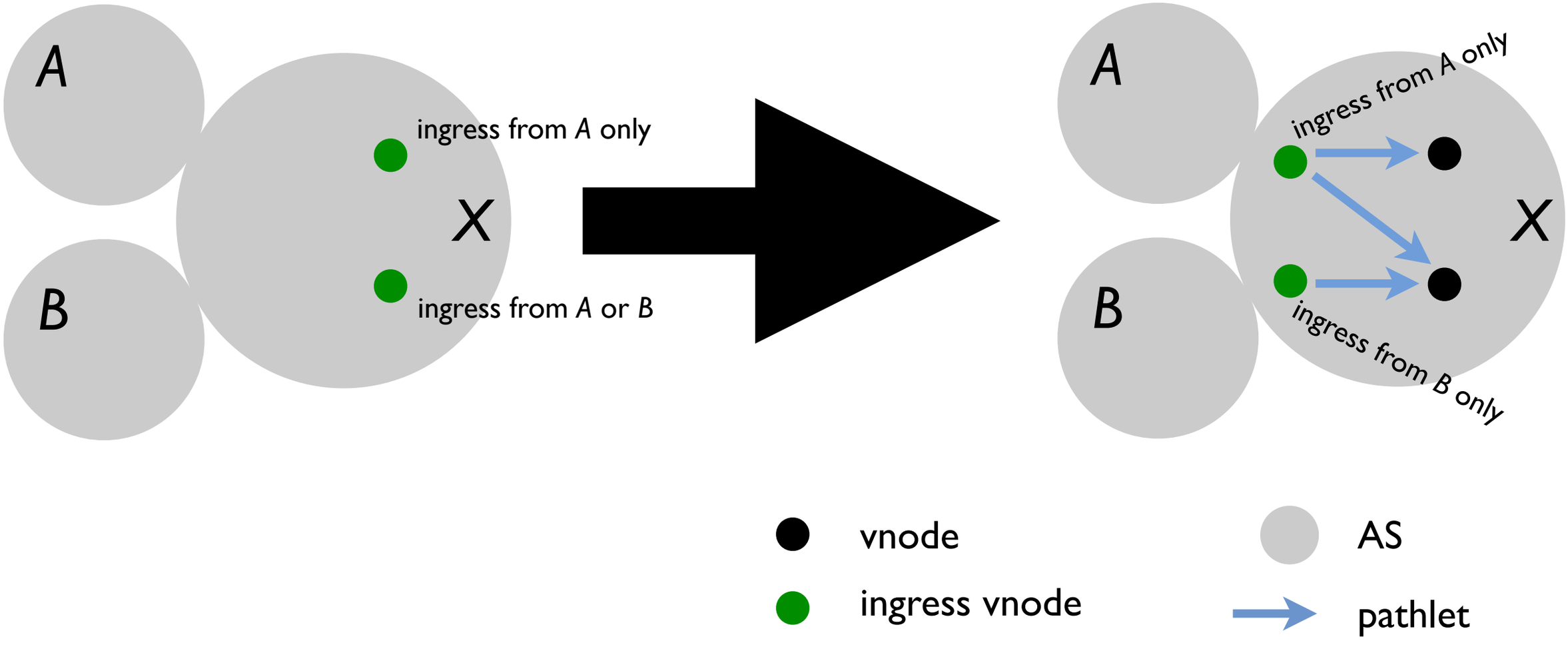
The issue is that tagging a packet with the intended vnode is slightly inconvenient. We felt it was cleaner to define the protocol so that packets contain only a list of pathlet identifiers (i.e., forwarding identifiers or FIDs). These are designed to be compact. Thus, the protocol spec in the paper says that a router specifies only a single ingress vnode for each neighbor.

Fortunately, it turns out that the single-ingress design is just as powerful as the set-of-ingress design, so the single-ingress protocol spec is fully capable of implementing VPNs. We take a look at this equivalence next.



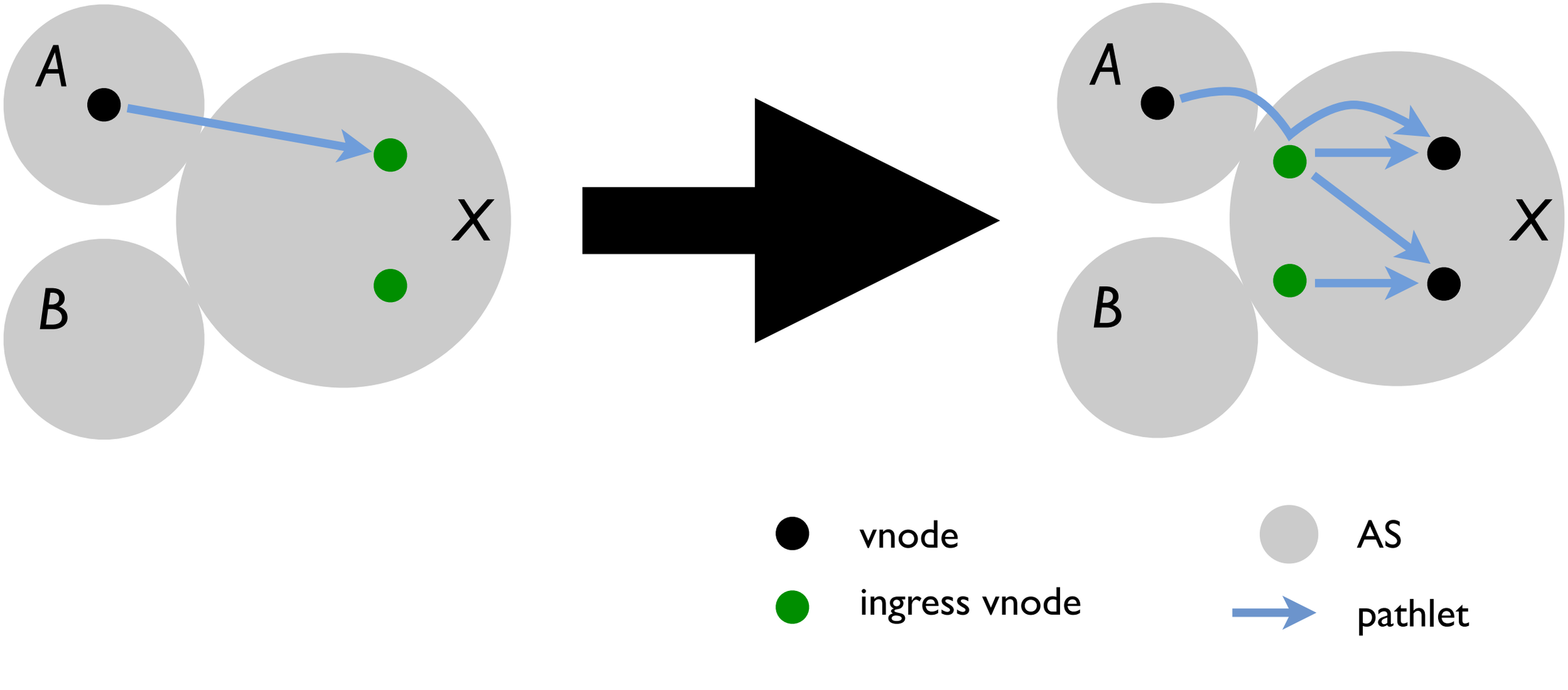
single ingress == multiple

It's easy to transform any multiple-ingress design to a single-ingress-per-neighbor design. We'll just show an example. Suppose *X* wants *A* to be able to ingress to both of *X*'s vnodes, but *B* should only ingress to one.



single ingress == multiple

One-hop pathlets into *X* are transformed into two-hop pathlets. Intuitively, instead of tagging a packet with its next-hop vnode, we're pushing another FID onto the front of the route.



VPN with one ingress

Putting it all together, here is a pathlet routing network which implements the desired VPN using only a single ingress vnode for each neighbor. The ingress vnode for a particular neighbor is, as you might expect, the one drawn closest to that neighbor.

This looks a bit complex, but in practice we would probably define the network as it looks in Slide 3 (which is more convenient for the operator) and then use the procedure of Slide 6-7 to automatically compile down to this single-ingress representation (which is more convenient for the data plane.)